APLICACIÓN PARA LA ELABORACION DE RECIBOS DE PAGO ÁREA TESORERÍA UNIVERSIDAD LIBRE SECCIONAL PEREIRA POR CONCEPTOS NO ESTABLECIDOS EN SIUL

JOSÉ JULIÁN RUIZ CORREA

CARLOS ALBERTO HUERTAS SUAREZ

Pereira

Universidad Libre Seccional Pereira

Facultad de Ingenierías

Programa de Ingeniería de Sistemas

2013

APLICACIÓN PARA LA ELABORACION DE RECIBOS DE PAGO ÁREA TESORERÍA UNIVERSIDAD LIBRE SECCIONAL PEREIRA POR CONCEPTOS NO ESTABLECIDOS EN SIUL

JOSÉ JULIÁN RUIZ CORREA

CARLOS ALBERTO HUERTAS SUAREZ

Presentación Proyecto de Grado para optar al título de

Ingeniero de Sistemas

Pereira

Universidad Libre Seccional Pereira

Facultad de Ingenierías

Programa de Ingeniería de Sistemas

2013

PÁGINA DE ACEPTACIÓN

Nota de aceptación:

_____
_____
_____
_____
_____

_____

Ing. Juan Manuel Cárdenas Restrepo

Coordinador Programa de Ingeniería de Sistemas

_____

Ing. Carlos Alberto Ocampo Sepúlveda

Asesor de Proyecto

_____

Ing. Daniel Aristizabal Torres

Director Investigaciones Facultad de Ingenierías

DEDICATORIA

Dedicamos este proyecto de grado a Dios, y a nuestros padres; a Dios porque ha estado con nosotros a cada paso que damos, cuidándonos y dándonos fortaleza para continuar. A nuestros padres, quienes a lo largo de nuestra vida han velado por nuestro bienestar y educación siendo nuestro soporte en todo momento, depositando su confianza inquebrantable en cada reto que asumimos sin vacilar ni un solo instante de nuestras virtudes.

José Julián Ruiz Correa

Carlos Alberto Huertas Suárez

A mi esposa Sandra Milena, mis hijos Daniel y el pequeño Nicolás que son el motor de mi vida, que me han llevado a superarme para brindarles un futuro más promisorio, por su paciencia y amor infinito durante cada noche que no les pude acompañar en sus juegos, en sus oraciones antes de dormir y en fin, porque debía cumplir una misión y una meta trazada para culminar con éxito mis estudios profesionales.

José Julián Ruiz Correa

A mis Padres Hugo Hernando (Fallecido), mi Madre Elsy Durley, mi hermana Durley Helena, mi Hijo Juan Sebastián y mi novia Ángela María que con su paciencia y tolerancia me han soportado en este tiempo en cual he sacrificado tiempo y los he descuidado un poco, pero con el tiempo se verá recompensado ese esfuerzo ya que por fin es otra meta alcanzada tanto personal como profesionalmente.

Carlos Alberto Huertas Suarez

AGRADECIMIENTOS

Queremos agradecer especialmente al Ingeniero Juan Manuel Cárdenas Restrepo coordinador del programa de Ingeniería en Sistemas por acompañarnos durante la realización de este proyecto y poner a nuestra disposición su tiempo y conocimiento; por habernos orientado y acompañado cada vez que debimos tocar una puerta y esta presentaba dificultad para abrirse. Porque de una u otra manera siempre estaba de un lado hacia el otro en cada una de las dependencias buscando apoyo para nosotros. Mil gracias.

Al Ingeniero Carlos Alberto Ocampo Sepúlveda asesor del proyecto por habernos brindado su acompañamiento y sus amplios conocimientos en el área revisándonos la ingeniería de software del proyecto y por el aporte de sus experiencias en el diario vivir, enseñándonos valores que se quedarán grabados en nuestras mentes y nos ayudarán a ser mejores profesionales y personas. Gracias por compartir con nosotras este paso tan importante de nuestras vidas.

A la Dra. Dora Patricia Ospina Parra Tesorera de la Universidad Libre Seccional Pereira por permitirnos el ingreso a su área para aplicar los conocimientos adquiridos durante nuestros estudios universitarios, facilitándonos la información necesaria para el desarrollo del proyecto y el acceso a su planta física y equipo de trabajo que nos acompañó amablemente y nos ayudó a lograr nuestras metas en este proyecto.

A nuestros profesores y compañeros que durante estos años de vida universitaria compartieron con nosotros sus conocimientos, nos enseñaron valores importantes para la vida como la amistad, el respeto y la lealtad, la alegría; nos acompañaron en la consecución de nuestras metas y en la vivencia de momentos irremplazables que se han de quedar por siempre en nuestros recuerdos.

TABLA DE CONTENIDO

Pág.

Pág.

LISTA DE ILUSTRACIONES

Pág.

Pág.

LISTA DE TABLAS

Pág.

GLOSARIO

API: (Application Programming Interface) o Interfaz de Programación de aplicaciones: Es un conjunto de funciones y procedimientos que permiten interacción entre diferentes aplicaciones.

BACKUP: Copia de respaldo de la información en cualquier medio de almacenamiento.

CACHÉ: Memoria utilizada por la unidad de procesamiento central (CPU) para almacenar información usada con frecuencia para acceder a ella rápidamente.

CASO DE USO: Herramienta de diagramación que permite visualizar el ambiente dentro del cual está enmarcado un sistema.

FIREBIRD: Base de datos SQL para desarrollo de aplicaciones locales y web.

FRAMEWORK: Estructura utilizada en el desarrollo de software que facilita a los programadores la realización de aplicaciones y operaciones complejas, encapsulándolas y automatizando los patrones utilizados para resolver las tareas.

HELPERS: son clases de ayuda para la realización de diferentes procesos en las cuales se agrupan funciones de uso común.

HERENCIA: Conceptos básicos de la programación orientada a objetos que permite ir de los más complejo a los más simple, consiste en la creación de una clase nueva a partir de una ya existente; esta nueva clase es denominada subclase y contiene los atributos y métodos de la clase principal con el fin de extender la funcionalidad de esta.

INTERFAZ DE USUARIO: es un medio compuesto por botones, menús, cuadros de dialogo, ventanas, formularios, teclado, mouse entre otros que le permiten al usuario comunicarse con un equipo de cómputo.

INTERFAZ GRÁFICA DE USUARIO: Tipo de interfaz de usuario que utiliza imágenes, objetos gráficos, iconos, menús entre otros para representar la información, brindándole al usuario una navegación fácil al momento de elegir comandos, seleccionar funcionalidades o ver listas de representaciones visuales.

LAYOUT: En el diseño de software muestra cómo deben estar organizados en cuanto a posicionamiento y tamaño los diferentes componentes de las interfaces de los usuarios; un layout se adapta automáticamente al tamaño de la ventana que se tenga en cada momento.

METALENGUAJE: Lenguaje utilizado para describir y especificar otro lenguaje (conocido como lenguaje objeto).

MULTIHILO: Es la ejecución de varias actividades de un programa en forma simultánea.

MULTIUSUARIO: Es la capacidad que tiene un programa de soportar simultáneamente varios usuarios que solicitan los mismos servicios o recursos.

MYSQL: Base de datos de código abierto de mayor aceptación mundial para la web. Es rápida, fiable y fácil de usar.

ORACLE: Sistema de gestión de bases de datos relacional de objetos. (Para mayor información este tema es ampliado en la página 67).

ORM (Object-Relational Mapping o Mapeo de Objetos a Bases de datos): Interfaz encargada de traducir la lógica de los objetos a la lógica relacional, permitiendo de esta manera acceder a la información almacenada en las bases de datos relacionales por medio de lenguajes de programación orientados a objetos.

POSTGRESQL: Sistema de gestión de bases de datos relacional orientado a objetos y libre, publicado bajo la licencia de software libre.

PROTOTIPO DE SOFTWARE: Representación inicial y limitada de las funcionalidades de un producto de software, que permite a las partes involucradas (clientes, usuarios y desarrolladores) explorar su uso y probar su funcionamiento.

SIUL: Sistema de información Universidad Libre.

TUPLAS: Una tupla sirve para agrupar, como si fueran un único valor, varios valores relacionados. El tipo de datos que representa a las tuplas se llama tuple.

TUPLE: Es una lista ordenada de n número de elementos o valores.

UML (Unified Modeling Language ó Lenguaje de Modelado Unificado): Lenguaje estándar utilizado para analizar, especificar y diseñar sistemas de software teniendo en cuenta ciertos requerimientos que definen el funcionamiento del mismo.

1. INTRODUCCIÓN

La Ingeniería de Sistemas constituye una herramienta fundamental de trabajo para aquellas personas que han tomado la decisión de crear y/o mejorar sistemas de información tendientes a la solución de problemas que tienen que ver con los procesos manejados en las empresas básicamente. Está representada por una metodología compuesta por un conjunto de etapas que se realizan de manera ordenada y escalonada para dar función a una aplicación en forma creciente y evolutiva.

Cada fase se integra por un conjunto de acciones orientadas a obtener productos determinados, tales como: especificaciones, diagramas, formatos, código, pruebas y documentos varios.

A continuación se presenta una solución de una aplicación para el área de tesorería de la universidad Libre de Pereira para uno de sus procesos específicos de atención al cliente.

Conforme se va avanzando en la elaboración del proyecto, se observará como todos los elementos de la Ingeniería de Sistemas se integran para dar con una solución acorde a la necesidad del área al mismo tiempo que se verificará la interacción de todos sus elementos que trabajen en acorde armonía para llegar a la solución final.

2. TÍTULO DE LA PROPUESTA

Aplicación para la elaboración de recibos de pago área tesorería Universidad Libre Seccional Pereira por conceptos no establecidos en Siul

3. ANTECEDENTES

Actualmente el área de Tesorería de la universidad Libre seccional Pereira y gracias a un desarrollo que hace unos años realizó el departamento de Sistemas, cuenta con un software para la elaboración de recibos de caja y este se sigue manteniendo; sin conocer ampliamente sus orígenes ya que no se sabe específicamente quien lo desarrollo, cuando, de qué manera y se pudo constatar que no existe ningún tipo de documentación al respecto, se puede deducir que en su momento no se tuvo en cuenta algunos aspectos formales ni técnicos tales como: sencillez, capacidad de actualización, utilidades para el aplicativo, copias de respaldo, códigos fuente, manuales técnicos, etc.

De acuerdo a la investigación adelantada con la encargada del área Dra. Dora Patricia Ospina Parra desde la creación de la mesa de ayudas hace aproximadamente 3 años, a través de este medio ha solicitado la mejora de la aplicación existente y su respectivo mantenimiento sin encontrar una respuesta satisfactoria y por consiguiente una solución definitiva a la problemática detectada.

Por este motivo y en aras de una mejor atención a los estudiantes se ha buscado fortalecer las actividades que ejecuta diariamente; convirtiéndose en una de las razones fundamentales para crear una nueva y moderna aplicación con un sistema automatizado y apropiado para satisfacer de manera adecuada y eficiente los requerimientos del personal encargado del área.

Por lo general un buen control de los recibos constituye una base sólida para la toma de decisiones mediante la observación de algunas estadísticas y de esta manera implementar nuevos controles para agilizar los procesos y prestar una excelente atención a todo el personal que se dirige a la dependencia para solicitar de sus servicios.

4. DESCRIPCIÓN DEL PROBLEMA

El problema presentado en el área de tesorería de la Universidad Libre de Pereira, nace de la necesidad que tiene la dependencia de contar con un buen aplicativo que de condiciones de respaldo, seguridad, control, automatización del proceso de elaboración e impresión de recibos de pago para estudiantes tanto en la Sede Belmonte como en la sede Centro que permite generar aquellos recibos que los estudiantes no tienen la posibilidad de elaborar a través de la plataforma SIUL tales como: Multas de biblioteca, pagos a cartera, abonos extraordinarios a cartera, certificados, pago copia de certificado de diploma, factura para consignación de cheque.

Se cuenta con un deficiente, desactualizado y condicionado programa que si bien inicialmente cubría las necesidades del área de forma limitada, con la apertura y avance de las nuevas tecnologías se le pueden dar nuevas perspectivas permitiendo reorganizar, agilizar y maximizar una serie de herramientas que permitan mayor eficiencia y eficacia para los usuarios encargados de su manipulación y administración.

Las limitaciones encontradas se enumeran a continuación en su respectivo orden de importancia.

- No existe un Back-Up o copia de seguridad y en caso de que el aplicativo o computador donde se encuentra instalado sufra algún tipo de daño de Hardware o Software que requiera ser formateado o reemplazado se perdería todas las utilidades que se tiene de este.

- No cuenta con la posibilidad de guardar un histórico de los recibos generados y por consiguiente de los consecutivos, impidiendo que el usuario pueda realizar algún tipo de consulta histórica o de un recibo en particular.

- La Universidad cuenta con la posibilidad de elaborar una factura para tres entidades bancarias diferentes a saber: Banco Davivienda, Bancolombia y Banco de Occidente, donde tiene sus productos financieros; pero el aplicativo únicamente expide recibos para el Banco de Occidente limitando que el estudiante pueda dirigirse a pagar a la entidad de su conveniencia y/o elección.

- El aplicativo que actualmente funciona para la dependencia, no tiene la posibilidad de ser actualizado, no cuenta con ningún tipo de documentación y no hay referencias de Manuales de usuario, Diccionarios de datos, Manuales técnicos por si el área de Sistemas quisiera retomarlo para su modificación, actualización o corrección.

- Si al momento de enviar una impresión por cualquier razón la impresora falla, el computador se bloquea o no es posible imprimir el recibo, este se pierde y el usuario debe volver a elaborar completamente la factura perdiendo tiempo y efectividad en su proceso.

Se debe tener en cuenta que el aplicativo para el manejo integral de recibos de pago área de tesorería es de vital importancia para la correcto funcionamiento del área de tesorería para la atención de estudiantes en el evento de facturación y el impacto que puede causar la inexistencia o daño permanente del mismo ocasionaría efectos de trascendencia al no tener un plan de contingencia o Back Up para facturar conceptos especiales y por consiguiente los estudiantes no puedan hacer sus pagos oportunamente.

Esta oficina presta sus servicios actualmente de lunes a sábado siendo permanente y constante los servicios ofrecidos por lo que es importante e imprescindible contar con una herramienta capaz de dar respuesta a la necesidad del área.

5. JUSTIFICACIÓN

La utilización de herramientas tecnológicas, pertinentes y actualizadas; para la administración y gestión de información, en el área de tesorería de la Universidad Libre seccional Pereira, facilita y mejora su realización; "Al hablar de tecnología la mayoría de las personas piensa en televisores, computadores, radios y otros. Estos sin duda son parte del acervo tecnológico de la humanidad, pero solo son representativos de finales del siglo pasado e inicios del presente. Si deseamos hablar de la tecnología en todo su alcance, tendríamos que definirla como todo aquel desarrollo humano que facilita sus labores y alivia sus necesidades"[1].

De acuerdo a lo que Jesús Ferro Bayona[2] expresa en su libro Educación y cultura, Colombia es un país que se encuentra atrasado en la aplicación y expansión de la tecnología hacia los lugares distantes de los centros urbanísticos y hacia otros sectores socioculturales como la educación; es por esto que llevar una solución tecnológica a los problemas de manejo gestión de información presentes hoy en el área de tesorería de la Universidad Libre seccional Pereira, aproxima a las personas que laboran en ella a las nuevas tecnologías de la información y las telecomunicaciones (NTICs), reduciendo de esta manera la brecha.

De la misma manera, hacer uso de herramientas digitales en lugar de materiales primarios y secundarios que deben extraerse del medio ambiente, tales como papel y lápices que incrementan la tala de árboles ó la fabricación de productos químicos que sirven como materia prima para la elaboración de tintas u otros insumos; ayuda en el cuidado y preservación del medio ambiente, lo cual hoy por hoy se ha convertido en una doctrina a seguir. "La producción masiva de papel supone la tala de árboles en una proporción de $3.2m^3$ de madera para la obtención de una tonelada de pasta de papel"[3].

---

[1] RODRÍGUEZ DURÁN, Armando, *et al*. Ciencia, tecnología y ambiente. Tercera Edición. Editorial Cengage ELT. p. 18.

[2] FERRO BAYONA. Op. cit., p. 201.

[3] VELÁZQUEZ DE CASTRO, F. y FERNÁNDEZ, Mará C. Temas de educación ambiental en las ciencias de la vida.

Así mismo, el uso de herramientas tecnológicas reduce los costos destinados por la institución para la consecución de materiales e insumos como papeles, marcadores, lápices; usados para la realización de ciertas labores administrativas que después se desechan cuando no serán nuevamente utilizados o cuando han perdido su funcionalidad. Además, cabe resaltar que hacer uso de herramientas novedosas, disminuye el tiempo empleado para la realización de estas actividades, optimizando de esta manera el nivel productivo de la organización. "En el crecimiento de la productividad influye la acumulación del capital físico y humano, así como la eficiencia en el uso de los recursos productivos"[4].

El estudio y proposición de una solución adecuada, de fácil acceso y tecnológica para la problemática encontrada en el área de tesorería de la Universidad Libre seccional Pereira, permite a los autores explorar, adecuar e implementar el conocimiento adquirido durante el transcurso de su carrera, utilizándolo para desarrollar herramientas servibles a la sociedad permitiéndoles mejorar su cotidianidad y calidad de vida. Según Fernando Doménech Betoret[5], cuando un estudiante aprende diferentes temas de memoria o de manera mecánica sin relacionarlos con conocimientos previos y práctica, se hace que este los asimile pero sin comprenderlos, manteniendo de esta manera el conocimiento adquirido aislado de su mente, ya que no puede aplicarlo a situaciones nuevas y por consiguiente llegará a olvidarlo en poco tiempo.

Por último, La Universidad Libre en el acuerdo No. 1 de 2005 por el cual se adopta y aprueba el Plan Integral de Desarrollo Institucional PIDI; dentro de los programas y proyectos del plan en su numeral 10, propende por una universidad con modernos apoyos tecnológicos y didácticos al servicio de la academia donde cuyo objetivo es evaluar y continuar el fortalecimiento de la provisión y uso de los medios informáticos en la labor académica.

Las políticas de la universidad incluyen el mejoramiento de sus procesos administrativos y financieros destacando la que dice "Una universidad moderna en su estructura, organización, planeación y financiación".

---

[4] GARRO BORDONARO, Nora; HERNÁNDEZ LAOS, Enrique y LLAMAS HUITRÓN, Ignacio. Productividad y mercado de trabajo. Primera Edición. 2000. México. p. 9.

[5] DOMÉNECH BETORET, Fernando. Proceso de enseñanza-aprendizaje universitario: aspectos teóricos y prácticos. Publicaciones de la Universitat Jaume. 1999. p. 121.

6. OBJETIVO GENERAL

Desarrollar un nuevo software tipo sistema de información que le permita al área de tesorería de la Universidad Libre de Pereira, la administración y gestión de recibos de pago por otros conceptos diferentes a los que se pueden expedir a través del sistema Siul.

6.1. OBJETIVOS ESPECIFICOS

- Realizar el levantamiento de requerimientos necesarios para el desarrollo del sistema de información, analizando los aspectos determinantes de los procesos administrativos en estudio y la forma en que estos restringen la solución ofrecida.

- Diseñar el prototipo del sistema de información que dará solución a la problemática.

- Implementar el prototipo del sistema de información que solucione los actuales problemas de administración y gestión de información, insumos y recursos presentes en el área de tesorería de la Universidad Libre de Pereira, empleando las herramientas tecnológicas seleccionadas y realizando las pruebas pertinentes.

- Elaborar los manuales técnico y de usuario para el prototipo de sistema de información implementado con el fin de brindarle a las personas involucradas los conocimientos, actitudes y habilidades que se requieren para lograr su manejo y desempeño óptimo.

7. HIPÓTESIS

Es posible desarrollar un software que controle los recibos, cree registros estadísticos, genere reportes y lleve un control apropiado de los documentos generados por cada estudiante. Para el desarrollo de este sistema se propone la utilización del software orientado a objetos, que servirá de apoyo ya que está basado en la planificación, análisis, diseño, e implementación de varias técnicas, incluyendo herencia, modularidad, seguridad e integridad y así se podrá definirlos y usarlos en el sistema.

8. DELIMITACION DEL PROYECTO

Este proyecto se realiza para el área de tesorería de la Universidad Libre Seccional Pereira. El tiempo disponible para su realización consta de veinticuatro semanas y comprende la etapa de planificación, análisis, diseño, evaluación, e implementación del software y la base de datos que complementara la información disponible en el programa SIUL de la universidad Libre.

Las limitantes que pueden afectar el normal desarrollo y puesta en marcha de la aplicación son limitantes de tiempo, recursos humanos y recursos físicos, ya que se debe contar con la disponibilidad de los funcionarios de la universidad con respecto al horario del que pueden disponer para atender las demandas contestar preguntas resolver interrogantes que se presentan y deban ser contestados; muchas veces porque por sus ocupaciones y/o jornadas laborales algunas veces no es posible contar con su atención y esto genera retrasos en la ejecución del mismo.

Por otra parte el acceso a los recursos físicos ya que se debe tener disponibilidad del servidor de pruebas para poder hacer las mismas, detectar errores, corregirlos y poner en marcha la aplicación en el servidor de tiempo real.

Cómo una limitante más, se tiene la necesidad de unos permisos escritos tales como acuerdo de confidencialidad, carta de presentación y autorización para que los estudiantes cuenten con el permiso para el acceso la información que se requiere.

El limitante final que se presentó para dejar el Software instalado y en pleno funcionamiento y utilidad, se debió a los permisos que no se pudieron conseguir porque el área legal y de sistemas considera de cierta manera la vulneración de su seguridad e integridad al permitir el acceso pleno a las bases de datos por lo que se debió crear una base de datos alterna en Oracle tal cual lo es la de SINU y hacer entrega del software que puede ser implementado cuando las áreas mencionadas así lo consideren.

9. MARCO REFERENCIAL

A. Marco Teórico

Para el desarrollo de un software que puede ser aplicado como una herramienta útil para mejorar la gestión del área de tesorería de la Universidad Libre Seccional Pereira, es necesario tener en cuenta que, en todo desarrollo de sistemas de Software es de vital importancia definir una metodología. Esta permite a los desarrolladores seguir alguna especificación en cada una de las etapas del desarrollo del sistema, desde los requerimientos iniciales hasta las pruebas finales, que haga que el mismo sea relacionado y además consecuente.

Explica porque se hace uso de la ingeniería de Software para el desarrollo de este proyecto y la utilización de las diferentes metodologías, herramientas y técnicas que la componen para de una manera ordenada y lógica se logre el desarrollo de este proyecto.

Como ya se ha documentado, la Universidad Libre de Pereira en el área de Tesorería, tiene como una de sus funciones la expedición de recibos de pago para estudiantes tanto en la Sede Belmonte como en la sede Centro que permite generar aquellos recibos que los estudiantes no tienen la posibilidad de elaborar a través de la plataforma SIUL tales como: multas de biblioteca, pagos de extemporaneidad, abonos extraordinarios, etc. Que cumpla con las condiciones de seguridad, agilidad, respaldo, facilidad, eficiencia, entre otras. Es por esto que se proyecta el desarrollo de un Software que se pueda aplicar como herramienta útil para la ejecución de la actividad propia de la dependencia de tesorería.

Se debe referir que en todo desarrollo de un sistema de este tipo, es imprescindible definir el tipo de metodología a emplear con el objetivo de que se puedan seguir ciertas especificaciones en cada etapa de ejecución de actividades desde el momento mismo en que se planteó la necesidad por parte de la dependencia, hasta la implantación final del producto de tal manera que este sea coherente y cumpla con todos los parámetros propios de la actividad de Software.

Para el modelado del proyecto de software, se ha establecido que se hará uso de la metodología UML que se puede ampliar claramente en [3] para el análisis y diseño. Se debe recordar que esta metodología es un lenguaje gráfico de modelado que permite visualizar, especificar, construir y documentar las funciones del sistema, permitiendo describir métodos o procesos claramente.

La utilización del lenguaje de modelado UML, el cual está compuesto por diversos elementos gráficos que se combinan para formar diagramas cuya finalidad es presentar diferentes perspectivas del sistema a las cuales se le conoce como modelo.

Es importante destacar que un modelo UML describe lo que supuestamente hará el sistema, pero no dice como implementarlo. En capítulos posteriores se describirán los diagramas del UML y los conceptos que representan.

Para dar solución al problema de Software, se puede seleccionar una herramienta de entre el variado grupo que existe, algunas basadas en procesos previamente establecidos otras en las acciones realizadas por el usuario, todas ellas dadas las actuales características que se desenvuelven en un ambiente gráfico y muy cercano al usuario. Es así como se ha seleccionado Visual Basic. Net, lenguaje que permite desarrollo orientado a objetos que viene siendo un complemento perfecto de UML en un ámbito gráfico que ha demostrado alta versatilidad en cuanto al desarrollo de aplicativos permitiéndose generar su propio código y acercando al usuario de una manera amigable además de simple a la realización de sus actividades. Este es un lenguaje que tiene más de veinticinco años de constante desarrollo; que en su comienzo era un lenguaje basado en código; motivo por el cual estuvo a punto de desaparecer. A comienzos de los años 90 tuvo su primera aproximación a un ambiente gráfico con el Q-BASIC momento en el cual volvió a entrar en el corazón de los programadores aficionados quienes lo llevaron a su primera versión realmente gráfica. A partir de este momento dejó de ser un juguete para principiantes para convertirse en una herramienta de desarrollo de software de calidad; tanto así que ha sido modelo para otros lenguajes posteriores.

Ingeniería de Software: Representa la principal herramienta de trabajo de los desarrolladores de sistemas de información. Está representada por una metodología compuesta por un conjunto de etapas que se realizan ordenadamente para dar vida a una aplicación en forma progresiva. Cada etapa se integra por un conjunto de acciones encaminadas para obtener productos específicos, como: especificaciones, diagramas, formatos, código, pruebas y documentos varios tales como manuales técnicos y de usuario.

Reingeniería: La reingeniería se produce en dos niveles distintos de abstracción. En el nivel de negocios, la reingeniería se concentra en el proceso de negocios con la intención de efectuar cambios que mejoren la competitividad en algún aspecto de los negocios. En el nivel del software la reingeniería examina los sistemas y aplicaciones de información con la intención de reestructurarlos o reconstruirlos de tal modo que muestren una mayor calidad.

La reingeniería de procesos de negocios (BPR) define los objetivos de negocios, identifica y evalúa los procesos de negocio ya existentes (en el contexto de los objetivos definidos), especifica y diseña los procesos revisados, y construye prototipos, refina e instancia esos procesos en el seno de un negocio. Al igual que la ingeniería de información, BPR suele ser la definición de formas en que las tecnologías de la información puedan prestar un mejor apoyo a los negocios [Pressman, 1998].

Es así que la re-ingeniería es el proceso de examinar un software, programa, existente y/o modificarlo con la ayuda de herramientas automatizadas para:

- Mejorar su futuro mantenimiento.

- Actualizar su tecnología.

- Extender su expectación de vida.

- Capturar sus componentes en un repositorio, donde las herramientas CASE (Computer-Aided Software Engineering) pueden ser utilizadas para mantenerlo.

- Incrementar su productividad de mantenimiento [McClure, 1992].

La reingeniería usualmente implica cambiar la forma, cambiar los nombres de los datos y sus definiciones, reestructurar los procesos lógicos, de un programa y mejorar su documentación. En este caso, la funcionalidad, comportamiento, del programa no cambia; sino, únicamente se modifica su forma. En otros casos, el proceso de reingeniería va más allá de la forma e incluye el rediseño cambiando la funcionalidad del programa para alcanzar los requerimientos del usuario.

Análisis: Es el proceso de examinar la cartera de sistemas existentes para entender mejor los componentes de los sistemas y como funciona el programa, para identificar los mejores candidatos para reingeniería, y para medir la calidad del sistema.

Reestructuración: Es el proceso de cambiar la forma del software, las definiciones y nombres de los datos y el código del programa, sin alterar su funcionalidad. El objetivo principal de la reestructuración es hacer el programa más fácil de entender.

Ingeniería inversa: Es el proceso de analizar un software, programa, para reconstruir la descripción de sus componentes y de la interrelación entre ellos. Una descripción de nivel superior del programa es recuperada de su nivel inferior,

forma física. El objetivo de la ingeniería inversa es re documentar el sistema y descubrir la información de diseño como una ayuda para incrementar el entendimiento del programa. Las herramientas de ingeniería inversa extraen información acerca de los datos, arquitectura y diseño de procedimientos de un programa ya existente.

Migración: Es el proceso de convertir un sistema computacional, programa, de un lenguaje a otro moviéndolo de un sistema operativo a otro, o actualizando su tecnología.

9.1 METODOLOGÍAS DE DESARROLLO DE SOFTWARE

Debido a la complejidad del proceso de desarrollo de software, actualmente se cuentan con diversas metodologías que van desde propuestas tradicionales centradas primordialmente en el control y estructuración de actividades, pasando por propuestas que se basan en el factor humano hasta basadas en el producto de software; esta variedad de metodologías es necesaria puesto que existen diferentes tipos de proyectos a los cuales se les debe implementar una metodología particular de acuerdo a sus características y requerimientos. Se pueden encontrar los siguientes tipos de metodologías.

9.1.1 Modelo lineal o en cascada. Es el modelo de desarrollo Cuenta con un proceso secuencial, en el cual se ordenan rigurosamente las etapas de definición, análisis, diseño y construcción, no se puede dar inicio a la siguiente etapa hasta que no se finalice la anterior. "Cada vez que finaliza una etapa se obtiene un documento o producto final, que revisado, validado y aprobado, sirve como aproximación y documentación de partida para la siguiente"[6].

Las etapas de esta metodología son labores importantes del desarrollo:

Definición de requerimientos: Es la fase inicial del proyecto, en esta se obtienen todos los elementos, metas, y requisitos del sistema a partir de consultas con los clientes y usuarios finales. El resultado de esta etapa es un documento donde se

[6] Barra BARRANCO DE AREBA, Jesús. Metodología del análisis estructurado de sistemas. Segunda Edición. Editorial Ortega. 2001. p. 44.

encuentran detallados todos los objetivos y las necesidades que debe cubrir el software a desarrollar.

Diseño del sistema y del software: En esta etapa se dividen el hardware y el software obteniendo una arquitectura completa del sistema, además a partir de la especificación de los requerimientos de la etapa anterior se realiza la selección de las herramientas, algoritmos y abstracciones fundamentales que en conjunto llevan a la solución del problema planteado.

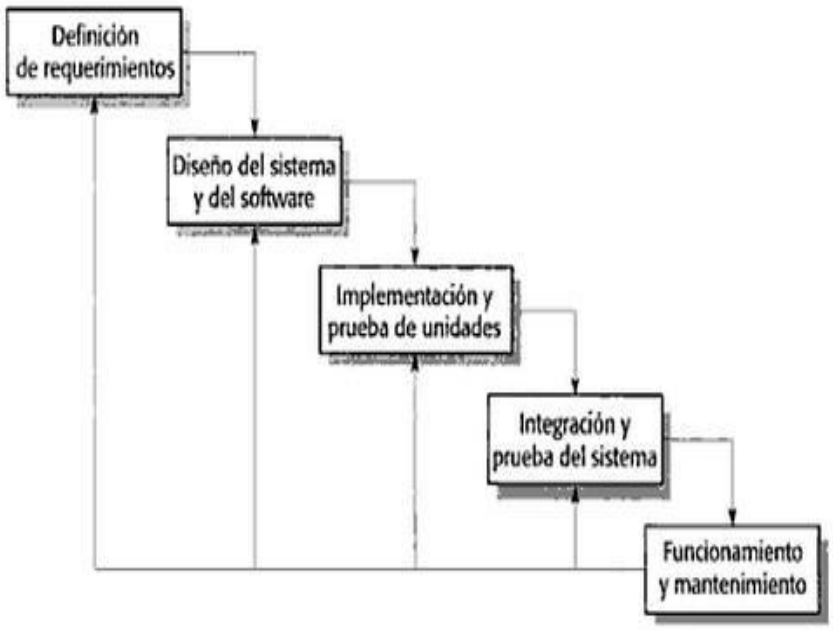
Implementación y prueba de unidades: En esta etapa se lleva a cabo el diseño del sistema a partir de la descomposición y organización de los elementos que lo conforman en unidades que puedan ser desarrolladas por separado, con el fin de aprovechar los beneficios del trabajo en equipo. Después se deben realizar pruebas para verificar que cada una de las unidades cumpla con los requerimientos establecidos.

Integración y pruebas del sistema: Las unidades individuales después de ser programadas se unifican y se prueban como un sistema completo antes de ser entregado al cliente. Para verificar que su funcionamiento es el correcto, cada funcionalidad del programa debe cumplir con los requerimientos definidos en las primeras etapas del desarrollo.

Después de comprobar que los resultados de las pruebas son consistentes con los requerimientos establecidos, el sistema es entregado al cliente.

Funcionamiento y mantenimiento: Generalmente es la fase más larga y critica del ciclo de vida del software, debido a que después de ser instalado y puesto en funcionamiento se pueden presentar problemas generados por un errónea definición de los requerimientos, por ende se deben realizar labores de mantenimiento al sistema, que incluyen tareas de corrección de errores, mejoramiento de unidades programadas y distinción de los servicios una vez sean encontrados y definidos nuevos requerimientos.

Ilustración 1. Modelo lineal o en Cascada



Fuente: SOMMERVILLE, Ian. Ingeniería del software. Séptima Edición. Editorial Pearson Education, S.A. 2005. 30

Las ventajas de implementar este modelo son:

- La información asociada al proyecto se mantiene bajo un orden específico permitiendo que no se mezclen las fases del ciclo de vida del software.

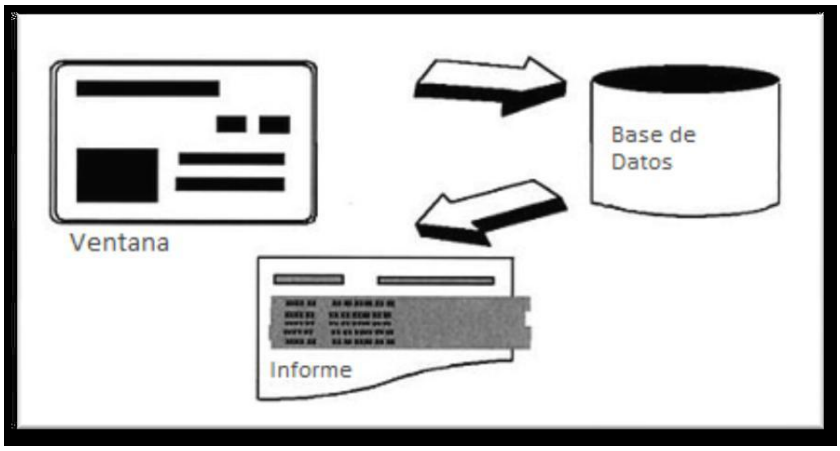
- El desarrollo de las fases puede ser realizado por equipos de trabajo diferentes, gracias a la detallada documentación que se debe generar al terminar cada una de las fases.

- Su implementación es sencilla debido a que sigue estrictamente el ciclo de vida del software.

Este modelo está orientado a proyectos de corto plazo, de poca innovación y proyectos definitivos y detallados, debido a que si desea realizar algún cambio en desarrollo será difícilmente asumido, además para visualizar el progreso del proyecto se debe esperar a que este se encuentre en una etapa muy avanzada.

9.1.2 Modelo Prototipado Se basa en la realización de prototipos, los cuales sufren constantes ajustes en todas las etapas del desarrollo, estos cambios están basados en las necesidades del usuario final, para ello se le muestra una vista preliminar de parte del software. Básicamente, funciona a prueba y error, ya que si al usuario no le gusta una parte del prototipo, esta debe ser replanteada hasta lograr que el usuario quede satisfecho.

Ilustración 2. Modelo Prototipado.



Fuente: BARRANCO DE AREBA, Jesús. Metodología del análisis estructurado de sistemas.

9.1.3 Modelo incremental o evolutivo Al inicio del proyecto, los clientes proveen un conjunto de requisitos estables a cumplir en la primera etapa, estos deben satisfacerse; las siguientes versiones del proyecto cumplen con los requisitos que faltan, de esta manera el sistema va evolucionando en cada una de sus fases. “Durante el desarrollo, se puede llevar a cabo un análisis adicional de requerimientos para los requisitos posteriores, pero no se aceptan cambios en los requerimientos para el incremento actual”[7].

9.1.4 Modelo de síntesis automatizado Este modelo se basa en el uso de una tecnología de software que genera código de forma automática a partir de la especificación de los requerimientos del sistema, para comenzar el desarrollo se

[7] SOMMERVILLE, Ian. Ingeniería del software. Séptima Edición. Editorial Pearson Education, S.A. 2005. p. 67.

debe entregar una definición detallada de la función del negocio, y luego esta se va refinando al detalle hasta que finalmente entrega el código que cumple con los requerimientos brindados por el cliente.

El inconveniente que presenta este modelo es que "el código generado precisa un modelo run-time para ejecutarse, y normalmente este módulo software debe estar instalado en todas aquellas estaciones de trabajo donde se precise ejecutar. Si se cuenta con numerosas estaciones, se tendrá un problema de despliegue y actualización del software"[8]**.**

Su ventaja ocurre al momento de realizar el mantenimiento del sistema, debido a que este se ejecuta sobre la especificación de los requerimientos, permitiendo tener una documentación actualizada y realizar incluso procesos de reingeniería.

9.1.5 Modelo en espiral Este modelo implementa una estrategia con el fin de minimizar los riesgos en momentos de incertidumbre, está compuesto por cuatro etapas básicas: **especificación**, en la cual se realiza la definición de las restricciones y los objetivos; **alternativas**, donde se analizan las posibles soluciones al problema planteado; **evaluación**, en esta etapa se determinan los riesgos y costes; y finalmente el ciclo de **desarrollo** se efectúa de manera lineal con la generación de productos.

De este modo, el modelo en espiral, enfatiza ciclos de trabajo analizando el riesgo de cada uno antes de comenzar el siguiente ciclo. "Cada ciclo comienza con la identificación de los objetivos, soluciones alterativas, restricciones asociadas con cada alternativa y, finalmente, se procede a su evaluación. Cuando se identifica incertidumbre, se utilizan diversas técnicas para reducir el riesgo de las distintas alternativas"[9].

Cada que se termina un ciclo del modelo en espiral, se realiza un análisis de los objetivos que fueron alcanzados y los planes para el siguiente ciclo.

---

[8] BARRANCO DE AREBA. Op. cit., p. 46.

[9] WEITZENFELD, Alfredo. Ingeniería Del software Orientada a Objetos Con Java E Internet. Editorial Cengage Learning. 2005. p.53.

Ilustración 3. Modelo en Espiral

Fuente: BARRANCO DE AREBA, Jesús. Metodología del análisis estructurado de sistemas.

9.2 METODOLOGÍAS ÁGILES DE DESARROLLO DE SOFTWARE

El desarrollo ágil del software es un marco de trabajo conceptual que fue promovido en febrero de 2001 después de una reunión en los estados unidos donde nace el término "Ágil" aplicado al desarrollo de software con nuevos enfoques metodológicos.

De esta reunión fueron miembros 17 expertos en ingeniería de software, los cuales definieron los valores y principios básicos que debe tener un sistema que se pretenda desarrollar implementando una metodología ágil y conservando la buena calidad de los productos; "En esta reunión se creó **The Agile Alliance**, una organización sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida fue el **manifiesto ágil**, un documento que resume la filosofía ágil"[10].

[10] CANÓS, José H, LETELIER, Patricio, y PENADES, María Carmen. Metodologías Ágiles en el Desarrollo de Software. Universidad Politécnica de Valencia. 2003. p. 2.

9.2.1 El manifiesto ágil Según el manifiesto se presentan cuatro valores a tener en cuenta en el desarrollo de software con buenas prácticas[11]:

- **Los individuos e interacciones del equipo de desarrollo por encima de los procesos y las herramientas.** El recurso humano es el principal factor de éxito de un proyecto de software. Es más importante construir un buen equipo que construir el entorno. Muchas veces se comete el error de construir primero el entorno y esperar que el equipo se adapte automáticamente pero es mejor crear el equipo y que éste configure su propio entorno de desarrollo en base a sus necesidades.

- **Desarrollar software que funciona por encima de conseguir una buena documentación.** La regla a seguir es no producir documentos a menos que sean necesarios de forma inmediata para tomar una decisión importante. Estos documentos deben ser cortos y centrarse en lo fundamental.

- **La colaboración del cliente por encima de la negociación de un contrato.** Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo, desde el comienzo hasta la culminación del proyecto. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.

- **Responder a los cambios por encima de seguir estrictamente un plan.** La habilidad de responder a los cambios que puedan surgir a los largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta.

Los valores anteriores inspiran los doce principios del manifiesto. Son características que diferencian un proceso ágil de uno tradicional. Los dos primeros principios son generales y resumen gran parte del espíritu ágil. El resto tienen que ver con el proceso a seguir y con el equipo de desarrollo, en cuanto a metas a seguir y organización del mismo.

[11] HERRERA URIBE, Eliécer y VALENCIA AYALA, Luz Stella. Del Manifiesto ágil sus valores y principios. En: Scientia Et Technica. Mayo, 2007. Año/vol. XIII, no. 34. Universidad Tecnológica de Pereira. Pereira. p. 381-385.

Los principios son:

I. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte valor al proyecto.

II. Dar la bienvenida a los cambios en los requerimientos, incluso los tardíos. Se aprovechan los cambios para que el cliente tenga ventaja competitiva.

III. Entregar frecuentemente software funcionando desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.

IV. Las personas del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.

V. Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para que consigan realizar su trabajo.

VI. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.

VII. El software funcionando es la medida principal de progreso.

VIII. Los procesos ágiles promueven un desarrollo sostenible. Los patrocinadores, desarrolladores y usuarios deberían ser capaces de mantener relaciones cordiales.

IX. La atención continua a la excelencia técnica y al buen diseño incrementa la agilidad.

X. La simplicidad es esencial.

XI. Las mejores arquitecturas, requerimientos y diseños surgen de los equipos organizados por sí mismos.

XII. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

9.2.2 Programación Extrema (XP) Es sin duda el tipo de metodología ágil más utilizado, se basa en el desarrollo reiterado donde el cliente tiene participación activa en niveles “Extremos”, buscando retroalimentación continua con el equipo desarrollador, el cual debe mantener buen trabajo en conjunto con el fin de mantener una constante y fluida comunicación que brinde simplicidad en las soluciones implementadas y que al momento del sistema necesitar cambios estos

sean realizados de forma ordenada y rápida sin generar retrasos en las entregas del proyecto. Este tipo de metodología es adecuada para proyectos con altos riesgos técnicos y que necesite cambios en el transcurso de su desarrollo, es decir que contenga requisitos imprecisos.

Se utilizaron dos de los valores de la programación extrema que fueron: la simplicidad y la comunicación, donde se simplificó el diseño para agilizar el desarrollo y facilitar el mantenimiento, esta es la manera de mantener el código simple a medida que crece; siendo la mejor manera de que las cosas funcionen. Así se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo. La simplicidad y la comunicación son extraordinariamente complementarias. Con más comunicación resulta más fácil identificar qué se debe y qué no se debe hacer.

9.2.3 Modelo SCRUM El modelo SCRUM es una metodología de desarrollo bastante simple, pero se requiere dedicación y trabajo duro, pues no cuenta con el seguimiento de un plan, sino con la adaptación continua de las circunstancias de evolución del proyecto.

SCRUM siendo una metodología ágil tiene características como:

- Es de carácter adaptable más que predicativo.
- Es más orientado a personas que a procesos.
- Implementa la estructura ágil de desarrollo incremental: basado en iteraciones y revisiones.

El control del proyecto emplea las siguientes prácticas de la gestión ágil:

- Revisión de las iteraciones.
- Desarrollo incremental.
- Desarrollo evolutivo.
- Auto-organización.
- Colaboración.

9.3 PROGRAMACIÓN ORIENTADA A OBJETOS (POO)

Para hablar de la programación orientada a objetos primero se debe entender el concepto de objeto. Un objeto es un conjunto de variables y funciones

relacionadas entre sí, que cuentan con un estado, un comportamiento definido y una identidad que los distingue de los demás objetos, son usados además para representar entidades del mundo real, y contienen toda la información necesaria para realizar su debido modelado. "El estado de un objeto está definido por el valor de ciertas variables internas al objeto"[12], las cuales pueden cambiar según la recepción de mensajes externos a él o a cambios internos dentro del objeto, igualmente, el estado define el comportamiento de los objetos que se percibe debido a los cambios que sufre al tener contacto con otros objetos y/o por las respuestas ante los mensajes que recibe.

La programación orientada a objetos es un modelo de programación que organiza las aplicaciones y programas en colecciones de objetos que interactúan entre sí, para resolver un problema determinado. Este paradigma se basa en el modelo de objetos, que a su vez se fundamenta en la implementación de siete capacidades, donde abstraer, encapsular, modularidad y jerarquizar son consideradas como principales mientras que tipo, concurrencia y persistencia son denominadas capacidades secundarias.

- **Abstraer:** es la capacidad de identificar los atributos principales de un objeto, los cuales definen sus cualidades con respecto a otros objetos, "la abstracción surge de reconocer las similitudes entre objetos, situaciones o procesos en el mundo real, y la decisión de concentrarse en esas similitudes e ignorar las diferencias"[13]. Al momento de ser definidas, se debe tener en cuenta cual es el objeto y cuáles son sus partes, su objetivo es conseguir a partir de un tema generalizado una visión global del mismo.

"La abstracción es la forma en que nuestra mente modela la realidad, formando los objetos. Por eso se crean objetos en los programas que simulen los comportamientos de los objetos del mundo real"[14]. Para ayudar a medir la calidad de las abstracciones se deben tener en cuentan los siguientes aspectos.

- Acoplamiento: disminuir las líneas de asociación entre distintas abstracciones

---

[12] VÉLEZ SERRANO, José F. *et al*. Diseñar y programar, todo es empezar: Una introducción a la programación orientada a objetos usando UML y Java. Universidad Rey Juan Carlos. Escuela Técnica Superior de Ingeniería Informática. Editorial Dykinson. 2010. p. 5.

[13] Ibíd., p. 6.

[14] CUEVA CHAMORRO, Diego. *et al*. Principios de la programación orientada a objetos. Universidad privada Antenor Orrego. Escuela de Ingeniería de computación y sistemas. 2007. p. 9.

- Cohesión: maximizar dentro de una abstracción su grado de asociación
- Suficiencia y completitud: brindar la información necesaria para que su ejecución sea eficiente
- Primitividad: las abstracciones deben tener procedimientos sencillos y básicos.

Encapsular: le brinda al programador la posibilidad de decidir qué información va a ser oculta y cuál va a dar a conocer al resto de objetos, por esta razón al momento de realizar algún cambio en la información, la persona encargada debe conocer las operaciones definidas para ese objeto. Su objetivo es evitar que un sistema dependa de la forma como se han implementado abstracciones de otra parte.

Cuando se implementa esta capacidad la información de los métodos de los objetos es presentada como interfaces públicas para que puedan ser solicitadas, y sus atributos o características como datos privados e inaccesibles desde los otros objetos. "De esta forma es difícil que un objeto pueda cambiar los atributos de otro, desde fuera de él"[15].

Además, al momento de un objeto solicitar acceso a los atributos de otro, este cuenta con métodos de acceso que le permiten al programador controlar la información a la que están accediendo los otros objetos, evitando resultados no deseados a partir de cambios inesperados en los atributos.

Modularidad: se basa en el principio de programación de divide y vencerás, consiste en subdividir un programa en un conjunto de sentencias lógicas llamadas módulos, cada uno con un propósito específico que al momento de ser integrados forman la solución a los requerimientos definidos por un programa determinado.

Implementar esta capacidad genera muchas ventajas al momento de realizar labores de mantenimiento, diseño o revisión; debido a que estas pueden ser ejecutadas de manera simultánea por diferentes personas; Además, el tiempo empleado para efectuar abstracciones disminuye puesto que al encontrase el programa organizado por módulos lógicos su búsqueda será más sencilla.

Jerarquizar: es la capacidad de organizar jerárquicamente las abstracciones de un programa según su naturaleza, formando árboles de herencia. Las más utilizadas son: la jerarquía de clases y la jerarquía entre objetos.

---

[15] Ibíd., p. 10.

- Jerarquía de clases: se definen relaciones de herencia en las cuales los elementos que heredan son más genéricos, mientras que los que se encuentran heredando tienen propiedades más especializadas.
- Jerarquía de objetos: se clasifica en relaciones de asociación y de dependencia. "Las relaciones de asociación establecen relaciones estructurales entre objetos de forma que se establece una conexión entre ellos"[16], es decir, se pueden construir nuevos objetos a partir de la asociación de otros objetos menores. Mientras que las relaciones de dependencia se dan cuando un objeto usa a otro de manera local, también son llamadas relaciones de uso.

Tipo: Es una caracterización precisa asociada a un conjunto de objetos[17] que al compartir la misma interfaz son denominados objetos del mismo tipo. La asociación de un tipo determinado a un objeto se conoce como tipado y este evita que se generen abstracciones diferentes que puedan llegar a dificultar su uso de una manera no prevista.

Concurrencia: esta característica brinda la oportunidad de ejecutar varios procesos que coexisten simultáneamente, compartiendo datos y recursos; se presenta generalmente en problemas que demandan el empleo de múltiples procesadores utilizados todos al mismo tiempo para resolver un problema determinado. Igualmente, existen otro tipo de problemas que requieren para su solución la implementación de diversos eventos paralelos.

Persistencia: permite que la existencia de un objeto perdure en el tiempo y en el espacio, es decir, su información se conserva más allá de la ejecución de un programa, además posibilita la recuperación de los datos con el fin de ser utilizados nuevamente.

Según su tiempo de vida los datos se pueden clasificar en:

- Expresiones: su tiempo de vida no sobrepasa una línea de código.
- Variables locales: depende de la vida de la función en la cual se definen y usan.
- Variables globales: su ciclo de vida dura mientras se ejecute un programa.
- Datos que sobreviven de una ejecución del programa a otra.
- Datos que persisten a una versión de un programa.
- Datos que subsisten a pesar que ya no existen los programas, los sistemas operativos o los ordenadores en los que fueron desarrollados.

---

[16] VÉLEZ SERRANO. Op. cit., p. 8.

[17] Ibíd., p. 11.

Los tres primeros tipos de datos son soportados por los lenguajes de programación, mientras que los tres restantes corresponden al ámbito de las bases de datos.

MODELO VISTA CONTROLADOR

Es un estándar de la arquitectura MVC (Modelo Vista Controlador) que define la organización independiente del **Modelo** (Objetos de Negocio), la **Vista** (interfaz con el usuario u otro sistema) y el **Controlador** (controlador de la aplicación).

De esta forma, se divide el sistema en tres capas donde, se tiene la encapsulación de los datos, la interfaz o vista por otro y por último la lógica interna o controlador.

Modelo

Contiene el núcleo de la funcionalidad (dominio) de la aplicación.
Encapsula el estado de la aplicación.
No sabe nada / independiente del Controlador y la Vista.

Vista

Es la presentación del Modelo.
Puede acceder al Modelo pero nunca cambiar su estado.
Puede ser notificada cuando hay un cambio de estado en el Modelo.

Controlador

Reacciona a la petición del Cliente, ejecutando la acción adecuada y creando el modelo pertinente

Para entender cómo funciona un modelo vista controlador, se debe entender la división a través del conjunto de estos tres elementos y como estos componentes se comunican unos con los otros y con otras vistas y controladores externos al modelo principal. Es importante saber que el controlador interpreta las entradas del usuario (tanto teclado como el ratón), enviado el mensaje de acción al modelo y a la vista para que se proceda con los cambios que se consideren adecuados.

Comunicación

El modelo, la vista y el controlador deben comunicarse de una manera estable los unos con los otros, de manera que sea coherente con las iteraciones que el usuario realizara. Como es lógico la comunicación entre la vista y el controlador es

bastante básica pues están diseñados para operar juntos, pero los modelos se comunican de una manera diferente, un poco más sutil.

B. Marco conceptual

9.4 BASES DE DATOS

La base de datos es un conjunto de información estructurada en registros y almacenada en un soporte electrónico legible desde un computador. Cada registro constituye una unidad autónoma de información que puede estar a su vez estructurada en diferentes campos o tipos de datos que se recogen en dicha base de datos.

Las bases son cualquier conjunto de datos organizados para su almacenamiento en la memoria de un computador, diseñado para facilitar su mantenimiento y acceso de una forma estándar. Los datos suelen aparecer en forma de texto, números o gráficos. Desde su aparición en la década de 1950, se han hecho imprescindibles para las sociedades industriales.

Los primeros sistemas de bases de datos aparecieron a finales de los años cincuenta. En este periodo, muchas compañías se fueron dando cuenta de que los primeros sistemas informáticos brindaban la posibilidad de aplicar soluciones de ordenamiento y consulta de datos más eficientes. Ceri y Pelagatti (1985), definen una base de datos como una colección de datos relacionados lógicamente, pero dispersos sobre diferentes sitios de una red de computadoras.

Los sistemas de bases de datos deben presentarse a los usuarios con una visión de los datos organizados en estructuras llamadas relaciones. Detrás de una relación puede haber cualquier estructura de datos compleja que permita una respuesta rápida a una variedad de consultas, aunque el usuario de un sistema no tiene que preocuparse por la estructura de almacenamiento que este presenta.

9.4.1. Elementos de una Base de Datos

- **Datos:** Es la parte esencial de la información, es decir, la información que llega a la base de datos.
- **Atributos:** Son los diferentes campos que conforman la estructura de una base de datos.
- **Campos:** Es la unidad más pequeña de datos.

- **Registro:** Es un conjunto de campos o atributos relacionados entre sí.
- **Archivo:** Es un conjunto de registros relacionados.

9.4.2. Modelos de Bases de Datos Además de la clasificación por la función de las bases de datos, éstas también se pueden clasificar de acuerdo a su modelo de administración de datos.

Un modelo de datos es básicamente una "descripción" de algo conocido como contenedor de datos (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores. Los modelos de datos no son cosas físicas: son abstracciones que permiten la implementación de un sistema eficiente de base de datos; por lo general se refieren a algoritmos, y conceptos matemáticos.

Algunos modelos con frecuencia utilizados en las bases de datos son:

Bases de datos jerárquicas En este modelo los datos se organizan en forma de árbol invertido (algunos dicen raíz), en donde un nodo padre de información puede tener varios hijos. El nodo que no tiene padres es llamado raíz, y a los nodos que no tienen hijos se los conoce como hojas.

Las bases de datos jerárquicas son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos permitiendo crear estructuras estables y de gran rendimiento.

Base de datos de red Éste es un modelo ligeramente distinto del jerárquico; su diferencia fundamental es la modificación del concepto de nodo: se permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico).

Bases de datos transaccionales Son bases de datos cuyo único fin es el envío y recepción de datos a grandes velocidades, estas bases son muy poco comunes y están dirigidas por lo general al entorno de análisis de calidad, datos de producción e industrial, es importante entender que su fin único es recolectar y recuperar los datos a la mayor velocidad posible, por lo tanto la redundancia y duplicación de información no es un problema como con las demás bases de datos, por lo general para poderlas aprovechar al máximo permiten algún tipo de conectividad a bases de datos relacionales.

Bases de datos relacionales Éste es el modelo utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los

modelos de base de datos. Su idea fundamental es el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas". Pese a que ésta es la teoría de las bases de datos relacionales creadas por Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar. Esto es pensando en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla).

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información.

El lenguaje más habitual para construir las consultas a bases de datos relacionales es SQL, Structured Query Language o Lenguaje Estructurado de Consultas, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

Durante su diseño, una base de datos relacional pasa por un proceso al que se le conoce como normalización de una base de datos.

Bases de datos orientadas a objetos Este modelo, bastante reciente, y propio de los modelos informáticos orientados a objetos, trata de almacenar en la base de datos los objetos completos (estado y comportamiento).

Una base de datos orientada a objetos es una base de datos que incorpora todos los conceptos importantes del paradigma de objetos:

- **Encapsulación:** Propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.
- **Herencia:** Propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases.
- **Polimorfismo**: Propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

En bases de datos orientadas a objetos, los usuarios pueden definir operaciones sobre los datos como parte de la definición de la base de datos. Una operación (llamada función) se especifica en dos partes. La interfaz (o signatura) de una operación incluye el nombre de la operación y los tipos de datos de sus argumentos (o parámetros). La implementación (o método) de la operación se especifica separadamente y puede modificarse sin afectar la interfaz. Los programas de aplicación de los usuarios pueden operar sobre los datos invocando

a dichas operaciones a través de sus nombres y argumentos, sea cual sea la forma en la que se han implementado. Esto podría denominarse independencia entre programas y operaciones.

Bases de datos deductivas Un sistema de base de datos deductiva, es un sistema de base de datos pero con la diferencia de que permite hacer deducciones a través de inferencias. Se basa principalmente en reglas y hechos que son almacenados en la base de datos. Las bases de datos deductivas son también llamadas bases de datos lógicas, a raíz de que se basa en lógica matemática. Este tipo de base de datos surge debido a las limitaciones de la Base de Datos Relacional de responder a consultas recursivas y de deducir relaciones indirectas de los datos almacenados en la base de datos.

9.4.3. Funciones de un Sistema de Gestión de Bases de Datos Los sistemas de Gestión de Base de Datos (DataBase Management System) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

Ullman & Widom (1997) establecen que las funciones básicas de un sistema de gestión de base de datos son:

1. Permitir a los usuarios crear nuevas bases de datos y especificar su estructura, utilizando un lenguaje o interfaz especializado, llamado lenguaje o interfaz de definición de datos.

2. Dar a los usuarios la posibilidad de consultar los datos y modificarlos, utilizando un lenguaje o interfaz apropiado, generalmente llamado lenguaje de consulta o lenguaje de manipulación de datos.

3. Permitir el almacenamiento de grandes cantidades de datos durante un largo periodo de tiempo, manteniéndolos seguros de accidentes o uso no autorizado y permitiendo un acceso eficiente a los datos para consultas y modificaciones.
4. Controlar el acceso a los datos de muchos usuarios a la vez, impidiendo que las acciones de un usuario puedan afectar a las acciones de otro sobre datos diferentes y que el acceso simultáneo no corrompa los datos.

Módulo: Es un pantalla de captura de información que se muestra en un monitor que en su conjunto desarrolla una acción determinada tal como captura, modificación consulta de datos, expedición de facturas entre otras.

Software: Hace referencia al programa o aplicativo intangible que se desarrolla para el área de tesorería.

Hardware: Son todos aquellos componentes físicos, tangibles que hacen parte de los sistemas de cómputo como tal se tienen: Monitor, teclado, torre. Impresora, scanner, etc.

La Base de Datos está estructurado bajo el sistema de gestión Oracle dada su soporte de transacciones, estabilidad, escalabilidad y soporte multiplataforma.

Hablando un poco acerca de Oracle, esta ha sido muy predominante para el mercado de servidores empresariales hasta hace poco; pero ha sufrido la competencia del Microsoft SQL Server y de la oferta de otros con licencia libre como PostgreSQL, MySQL o Firebird.

Es un manejador de base de datos relacional que hace uso de los recursos del sistema informático en todas las arquitecturas de hardware, para garantizar su aprovechamiento al máximo en ambientes cargados de información.
Es el conjunto de datos que proporciona la capacidad de almacenar y acudir a estos de forma recurrente con un modelo definido como relacional. Además es una suite de productos que ofrece una gran variedad de herramientas [4].

9.5. VISUAL STUDIO 2012

Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de programación entre los que se nombran: Visual C++, Visual C#, Visual J#, y Visual Basic .NET, al igual que entornos de desarrollo web como ASP.NET. Últimamente se han desarrollado las extensiones necesarias para muchos otros lenguajes.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión .NET 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

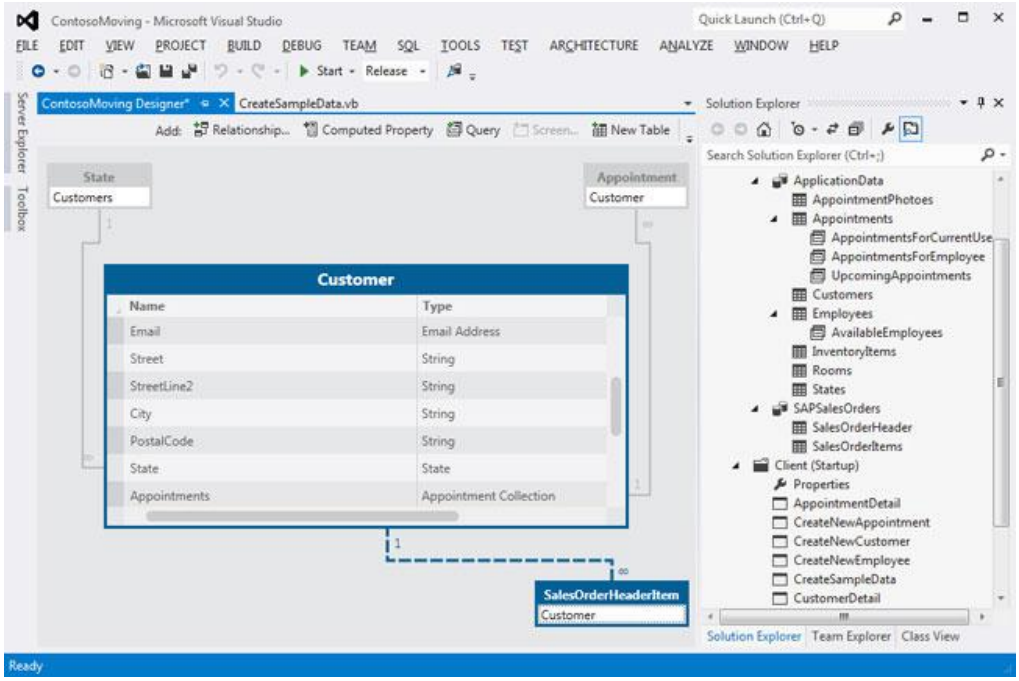
Visual Studio 2012 es un conjunto de herramientas de desarrollo y de otras tecnologías basado en componentes para crear aplicaciones eficaces de alto rendimiento. Además, Visual Studio está optimizado para diseño basado en equipos, desarrollo e implementación de soluciones empresariales.

Proporciona eficaces herramientas y servicios que permiten crear una nueva generación de aplicaciones, o bien modernizar las existentes. Los usuarios disfrutarán de una experiencia sin igual en distintas pantallas y dispositivos, siempre conectados a los servicios y datos que necesitan.

9.5.1.Novedades de Visual Studio 2012 Como bien se sabe, la nueva era de las aplicaciones modernas se encuentra en auge. Con dispositivos conectados y servicios basados en la nube, Visual Studio 2012 tiene más y mejores oportunidades. Cuando se va a realizar un desarrollador puede conectarse desde cualquier lugar, generar aplicaciones y ponerlas a disposición de los usuarios. Los equipos grandes y ágiles pueden darles a sus empresas ventajas significativas: cuánto más rápida sea la ejecución, mayor podrá ser la ventaja.

Por eso, Visual Studio 2012 es una de las versiones más importantes hasta el momento. Viene especialmente diseñado para prosperar en un entorno en el que las ideas escasean y la velocidad es fundamental. Estas son algunas formas en las que puede ayudarlo a convertir ideas en aplicaciones de forma rápida.

Ilustración 4. Entorno de Desarrollo



Con solo abrir el IDE, se puede notar que la interfaz permite simplificar el flujo de trabajo y brindar fácil acceso a las herramientas de uso cotidiano. Las barras de herramientas están simplificadas, permite la reducción de desorden de pestañas y tiene formas nuevas y rápidas de encontrar códigos. Todo esto permite facilitar la

navegación por su aplicación, para que se pueda trabajar de forma fácil. Para ampliar los conceptos acerca de Visual Studio consultar en [5].

9.6. CÓDIGO DE BARRAS

El código de barras es un código basado en la representación mediante un conjunto de líneas paralelas verticales de distinto grosor y espaciado que en conjunto contienen una determinada información, es decir, las barras y espacios del código representan pequeñas cadenas de caracteres. De este modo, el código de barras permite reconocer rápidamente un artículo de forma única, global y no confusa en un punto de la cadena logística y así poder realizar inventario o consultar sus características asociadas. Actualmente, el código de barras está estableciendo de manera masiva a nivelo mundial

La correspondencia o mapeo entre la información y el código que la representa se denomina simbología. Estas simbologías pueden ser clasificadas en grupos de acuerdo a dos criterios diferentes:

- **Continua o discreta:** en las simbologías continuas los caracteres comienzan con un espacio y en el siguiente comienzan con una barra (o viceversa). Sin embargo, en las simbologías discretas los caracteres comienzan y terminan con barras y el espacio entre caracteres es ignorado y generalmente de poca anchura.

- **Bidimensional o multidimensional:** En las simbologías bidimensionales las barras pueden ser anchas o estrechas. Sin embargo, las barras en las simbologías multidimensionales son múltiplos de una anchura determinada (X). De esta forma, se emplean barras con anchura X, 2X, 3X, y 4X.

Nomenclatura básica

- Módulo: Es la unidad mínima o básica de un código. Las barras y espacios están formados por un conjunto de módulos.
- Barra: El elemento oscuro dentro del código. Se hace corresponder con el valor binario 1.
- Espacio: El elemento claro dentro del código. Se hace corresponder con el valor binario 0.
- Carácter: Formado por barras y espacios. Normalmente se corresponde con un carácter alfanumérico.
- P: prefijo GS1 (por ejemplo, el número 770 - 771 corresponde a Colombia)

- Código de empresa: código asignado a las empresas registradas (5 - 8 dígitos)
- Código de producto: dígitos en blanco para el propietario de la marca
- C: dígito de control.

Situación en el producto

Los códigos de barras se imprimen en los envases, embalajes o etiquetas de los productos. Entre sus requisitos básicos se encuentran la visibilidad y fácil legibilidad por lo que es necesario un correcto contraste de colores. En este sentido, el negro sobre fondo blanco es el más habitual encontrando también azul sobre blanco o negro sobre marrón en las cajas de cartón ondulado. El código de barras lo imprimen los fabricantes (o, más habitualmente, los fabricantes de envases y etiquetas por encargo de los primeros) y, en algunas ocasiones, los distribuidores.

Para no entorpecer la imagen del producto y sus mensajes promocionales, se recomienda imprimir el código de barras en lugares discretos tales como los laterales o la parte trasera del envase. Sin embargo, en casos de productos pequeños que se distribuye individualmente no se puede evitar que ocupe buena parte de su superficie: rotuladores, barras de pegamento, entre otros.

Tipos de códigos de barras

Simulación de un código EAN-128Los códigos de barras se dividen en dos grandes grupos: los códigos de barras lineales y los códigos de barras de dos dimensiones.

Códigos de barras lineales

- EAN
- Code 128
- Code 39
- Code 93
- Codabar

Ilustración 5. Ejemplo de un Código de Barras.

De conformidad con el código de Barras en las facturas para la Universidad, este se entiende como un servicio que permite recaudar en oficinas y Centros de Pago y Recaudo de los Bancos, la cartera y otros conceptos de los estudiantes mediante la captura de un código que puede contiene la identificación de la Universidad (IAC), las referencias, el valor de pago y la fecha límite de pago.

Para la aplicación desarrollada el código IAC asignado a la Universidad Libre por la GS1 Colombia es:

- 9998007420

Mientras que para las facturas emitidas directamente por Sinú, el código IAC es:

- 9998007413

9.7. ¿QUÉ ES LA GS1 COLOMBIA?

GS1 Colombia, organización miembro de la red mundial GS1, brinda soluciones de conectividad a las empresas de diferentes sectores del país haciendo visible la información de sus productos a través de estándares de identificación y comunicación.

Es una organización sin ánimo de lucro que a través de su sistema de identificación donde el más conocido y punto de partida para múltiples soluciones, es el código de barras que proporciona un marco que permite que los productos, servicios, y la información, se muevan de manera eficiente y segura para el beneficio de las empresas y el mejoramiento de la vida de los clientes todos los días, y en todas partes. Los estándares GS1 aseguran intercambios eficaces entre las empresas, y actúan como guías básicas que facilitan la interoperabilidad y proporcionan estructura para muchas industrias.

Fue creada en 1988 por empresarios colombianos de Almacenes Éxito, Cafam y Unilever entre otros, con el propósito de trabajar en conjunto y encontrar mayor competitividad para sus organizaciones y por ende para la economía del país. Con base en el conocimiento de sus necesidades y requerimientos, estas empresas forman parte activa del desarrollo de estándares y soluciones que impactan directamente la eficiencia de sus sectores y beneficien a los consumidores. Team Foods, Cafam, Grandes Superficies de Colombia, Galletas Noel S.A.S, Fenalco, Carvajal Soluciones, Olímpica y Unilever, son las empresas que actualmente hacen parte del Consejo Directivo de GS1 Colombia.

Durante los últimos 25 años GS1 Colombia se ha enfocado en brindar beneficios a sus miembros y clientes que corresponden a más de 23.000 empresas a través del desarrollo de Redes de Valor, basados en la colaboración entre socios de negocio y la implementación de dichos estándares.

10. ANÁLISIS DE LOS REQUERIMIENTOS DE LA APLICACIÓN

Con el objetivo de recolectar la información necesaria para el desarrollo de este proyecto para el área de Tesorería de la Universidad Libre Pereira, se realizó una serie de entrevistas a la Tesorera y a la auxiliar de tesorería para posteriormente conocer y establecer los requisitos funcionales y no funcionales que debe tener la aplicación.

10.1 ENTREVISTA

Tabla 1. Formato de Entrevista. Tesorera

ENCUESTA SOBRE EL PROCESO DE EXPEDICIÓN DE UNA FACTURA DE PAGO SOLICITADA POR UN ESTUDIANTE EN EL ÁREA DE TESORERIA			No. 1
ENCUESTADOR:	**José Julián Ruiz Correa Carlos Alberto Huertas Suárez**	**FECHA:**	**Mayo 25 de 2013**
ENCUESTADA:	**DORA PATRICIA OSPINA PARRA**	**CARGO:**	**TESORERA**
PREGUNTAS			
1.	¿Con respecto al proceso de expedición de recibos que consideración tiene usted en cuanto a la aplicación que actualmente usan?		
	R= Pienso que es muy obsoleta y le hace falta funciones.		
2.	¿En el momento que usted requiera hacer algún tipo de consulta histórica o de control de alguna generación de recibo a través de que herramienta puede hacerlo?		
	R= No tengo ninguna herrmienta porque el aplicactivo actual no guarda absolutamente nada de lo que se hace con él		
3.	¿Que sucede si llegará a fallar el equipo donde actualmente tienen instalado el aplicativo o se dañara la aplicación como tal?		
	R= Nos quedaríamos sin con que elaborar recibos porque no existe ninguna copia u otro equipo soporte donde se tenga dicha aplicación		
4.	¿En el momento de expedir una factura a un estudiante considera usted que la aplicación es sencilla de usar?		
	R= Aunque es sencilla de usar, es muy manual, por que casi todos los datos se deben escribir manualmente en lugar de que sean capturados automáticamente. Se debe dirigir a otras pantallas y/o listados para extraer datos necesarios.		
5.	Hemos notado que si la persona escribe un dato errado, o la impresión por alguna razón se daña, no tienen la capacidad de retomar la información o el recibo ya elaborado y deben hacerlo de nuevo.		
	R= Aquí frecuentemente nos equivocamos con los datos diligenciados y/o las impresiones se dañan al atascarse el papel, entonces la auxiliar se ve en la obligación de hacer de nuevo el recibo porque no queda ningún registro.		
6.	Que tipo de control tienen con respecto a la validación de usuarios en el sistema, como saber quien elaboró un recibo respectivamente		
	R= No tenemos forma de llevar ningún control porque el sistema no tiene ninguna opción		
7.	¿En el ejercicio que acabamos de hacer imprimiendo una factura de prueba, nos damos cuenta que esta solamente se puede pagar en el banco de Occidente y los demás bancos no. Porque?		
	R= El aplicativo solamente expide los recibos para este banco no sabemos porque		

Continuación: Tabla 1. Formato de Entrevista 1. Tesorera

PREGUNTAS CERRADAS		SI	NO	NO SABE
8.	¿Cree usted importante que se deba hacer una validación de usuarios y perfiles en el sistema para el ingreso y así tener mayor control?	X		
9.	¿Considera necesario que de ahora en adelante lleve un consecutivo la expedición de facturas?	X		
10.	¿Opina que el cambio de la aplicación por una moderna y con mayores funcionalidades mejoraría la atención del área?	X		
11.	¿Considera que la modificación de la aplicación a un ambiente más amigable y fácil de usar trae mayores beneficios de uso?	X		

Tabla 2. Formato de Entrevista. Auxiliar Tesorería

ENCUESTA SOBRE EL PROCESO DE EXPEDICIÓN DE UNA FACTURA DE PAGO SOLICITADA POR UN ESTUDIANTE EN EL ÁREA DE TESORERIA			No. 1
ENCUESTADOR:	José Julián Ruiz Correa Carlos Alberto Huertas Suárez	FECHA:	Mayo 25 de 2013
ENCUESTADA:	SINDY NATALIA AGUDELO	CARGO:	AUXILIAR TESORERIA

PREGUNTAS	
1.	¿Con respecto al proceso de expedición de recibos que consideración tiene usted en cuanto a la aplicación que actualmente usan?
	R= Es lenta y cuando elaboro un recibo y se me daña o me equivoco debo volver a hacerlo de nuevo
2.	¿En el momento que usted requiera hacer algún tipo de consulta histórica o volver a mirar algún recibo a través de que herramienta puede hacerlo?
	R= No tenemos ninguna herrmienta que lleve un registro o guarde esta información
3.	¿Que sucede si llegará a fallar el equipo donde actualmente tienen instalado el aplicativo o se dañara la aplicación como tal?
	R= Nos quedaríamos sin el aplicativo de recibos porque no existe ninguna copia u otro equipo back-up
4.	¿En el momento de expedir una factura a un estudiante considera usted que la aplicación es sencilla de usar?
	R= Si es de fácil manipulación, pero todos los datos se deben capturar manualmente. Adicionalmente para mirar los conceptos de pago, si no los conocemos de memoria debemos abrir un archivito donde se tienen guardados
5.	Hemos notado que si la persona escribe un dato errado, o la impresión por alguna razón se daña, no tienen la capacidad de retomar la información o el recibo ya elaborado y deben hacerlo de nuevo.
	R= Es muy común que suceda algún problema con la elaboración de recibos y aveces las impresiones salen malas así que debo hacer de nuevo el recibo porque toda la información se pierde
6.	Que tipo de control tienen con respecto a la validación de usuarios en el sistema, como saber quien elaboró un recibo respectivamente
	R= No tenemos forma de llevar ningún control solamente con el horario y saber quien trabajó en ese horario
7.	¿En el ejercicio que acabamos de hacer imprimiendo una factura de prueba, nos damos cuenta que esta solamente se puede pagar en el banco de Occidente y los demás bancos no. Porque?
	R= El aplicativo solamente expide los recibos para este banco no sabemos porque

Continuación: Tabla 2. Formato de Entrevista 2. Auxiliar tesorería

PREGUNTAS CERRADAS		SI	NO	NO SABE
8.	¿Cree usted importante que se deba hacer una validación de usuarios y perfiles en el sistema para el ingreso y así tener mayor control?	X		
9.	¿Considera necesario que de ahora en adelante lleve un consecutivo la expedición de facturas?	X		
10.	¿Opina que el cambio de la aplicación por una moderna y con mayores funcionalidades mejoraría la atención del área?	X		
11.	¿Considera que la modificación de la aplicación a un ambiente más amigable y fácil de usar trae mayores beneficios de uso?	X		

10.2 REQUERIMIENTOS FUNCIONALES Y NO FUNCIONALES

10.2.1 Requerimientos no Funcionales

Tabla 3. Requerimiento no Funcional Usabilidad

RNF - 01	Usabilidad
VERSIÓN	1.0 (Agosto de 2013)
SUGERIDOPOR:	Desarrolladores - Personal Tesorería
DESCRIPCIÓN:	● El Software debe ser fácil y entendible, para ser usado por cualquier persona que tenga o no conocimientos en aplicaciones computacionales, y garantizar que el entrenamiento en el uso de la aplicación sea sencillo para que estos se adapten fácilmente a ella. ● El Sistema debe tener una interfaz gráfica con colores que sean agradables a la vista de los usuarios.

Tabla 4. Requerimiento no Funcional Portabilidad

RNF - 02	Portabilidad
VERSIÓN	1.0 (Agosto de 2013)
SUGERIDOPOR:	Desarrolladores
DESCRIPCIÓN:	● Para el Sistema a desarrollar se ha utilzado la plataforma Visual Studio 2012 que es compatible con la gran mayoría de Sistemas Operativos Windows tales como: Windows 8, Windows 7, Windows Vista, Windows XP, y para sistemas servidores Windows Server 2012, 2008, 2003 y anteriores. ● No requiere característica de Hardware adicionales

Tabla 5. Requerimiento no Funcional Seguridad

RNF - 03	Seguridad
VERSIÓN	1.0 (Agosto de 2013)
SUGERIDOPOR:	Desarrolladores - Personal Tesorería
DESCRIPCIÓN:	● El Sistema debe ser íntegro para evitar que los datos sean destruidos o alterados por personas ajenas al área de tesorería. Confiablepara garantizar el acceso a los datos solo a personas autorizada para cuyo fin cada uno tendrá su respectivo usuario y contraseña y deberá estar dsiponible los 7 dís de la semana 24 horas.

Tabla 6. Requerimiento no Funcional. Escalabilidad

RNF - 04	Escalabilidad
VERSIÓN	1.0 (Agosto de 2013)
SUGERIDOPOR:	Desarrolladores - Personal Tesorería
DESCRIPCIÓN:	● El sistema debe permitir la creación, eliminación o modificación de funcionalidades en el futuro, razones por las cuales se deja debidamente documentado con sus manuales técnicos y su código fuente.

Tabla 7. Requerimiento no Funcional. Mantenimiento

RNF - 05	Mantenimiento
VERSIÓN	1.0 (Agosto de 2013)
SUGERIDOPOR:	Desarrolladores - Personal Tesorería
DESCRIPCIÓN:	● El Software estará documentado completamente, con el fin de facilitar labores de mantenimiento, capacitación o mejoras futuras.

10.2.2 Requerimientos Funcionales

Tabla 8. Requerimientos Funcionales

FORMATO DE TRABAJO DE CAMPO		
TIPO		**FUENTE**
Observación Directa		Dora Patricia Ospina Sindy Natalia Agudelo
ANALISTAS	**FECHA**	**TIEMPO**
José Julián Ruiz Correa Carlos Alberto Huertas Suárez	Mayo 11 de 2013	4 Horas

No.	Lista de Requerimientos
1	La solución debe permitir el acceso a los siguientes perfiles de usuario, los cuales hacen parte del equipo de trabajo del área de Tesorería. Tesorera / Auxiliar Tesorería
2	Para poder acceder a la información consignada en la aplicación se debe asignar a los usuarios Tesorera/Auxiliar tesorería una cuenta de usuario y una contraseña que son validadas por el sistema en el momento de ingreso.
3	La Tesorera tiene la posibilidad de realizar: Consulta de estudiantes, Consulta de reportes
4	La Auxiliar de tesorería tiene la posibilidad de realizar: Consulta de estudiantes, Consulta de reportes y Expedición de facturas
5	El sistema debe permitir al usuario encargado de expedición de facturas la elección del Banco donde el estudiante desea realizar su pago
6	El sistema debe permitir la impresión física de los documentos que son generados por la aplicación
7	El sistema debe permitir a los usuarios salir correctamente de la aplicación.

11. INGENIERÍA DEL SOFTWARE

La Ingeniería de Software es la parte de la ingeniería que crea y mantiene las aplicaciones de software, utilizando tecnologías y prácticas de las ciencias computacionales, administración de proyectos, temas de ingeniería, el ámbito de la aplicación, y otros campos.

Es el desarrollo, operación y mantenimiento del software de forma sistemática, disciplinada y cuantificable, y el estudio de dichos métodos.

En otras palabras, es el estudio dedicado a la creación de software de buena calidad, barato y fácil de desarrollar y mantener.

Comienza a establecerse a finales de la década de 1960. Con el transcurso de los años se han desarrollado recursos que conforman la ingeniería del software, es decir, herramientas y técnicas de especificación, diseño e implementación del software.

La utilización de los recursos depende de la magnitud del proyecto, de la empresa a cargo, la experiencia de los desarrolladores, el financiamiento con el que se cuenta, entre otros.

El ingeniero de software se encarga de toda la gestión del proyecto para que éste se pueda desarrollar en un plazo determinado y con el presupuesto con que se cuenta

Incluye el análisis previo de la situación, el diseño del proyecto, el desarrollo del software, las pruebas necesarias para confirmar su correcto funcionamiento y la implementación del sistema. Para conocer en detalle se puede observar en [9].

11.1 ARQUITECTURA DEL SOFTWARE

La arquitectura del sistema en general, es la planeación, ya sea a nivel de infraestructura de red y hardware, o de software.

La arquitectura de software es el diseño de componentes de una aplicación (entidades del negocio), generalmente utilizando patrones de arquitectura. El diseño arquitectónico debe permitir visualizar la interacción entre las entidades del negocio y además poder ser validado, por ejemplo por medio de diagramas de

secuencia. Un diseño arquitectónico describe en general el cómo se construirá una aplicación de software. Para ello se documenta utilizando diagramas. Para este caso en particular se hará uso de los siguientes diagramas que serán descritos y elaborados en las páginas siguientes:

- **Diagrama de Caso de Uso**
- **Diagramas de Secuencia**
- **Diagramas de Clase**
- **Diagramas de base de datos** (Se verá en detalle en el capítulo 12 en la página 71)

Siendo los dos primeros los mínimos necesarios para describir la arquitectura de un proyecto que iniciará a ser codificado.

Depende del alcance del proyecto, complejidad y necesidades, el arquitecto elige qué diagramas elaborar.

11.2 DIAGRAMAS DE CASOS DE USO

Generalización de los actores Un diagrama de caso de uso es una herramienta que permite visualizar el ambiente dentro del cual está enmarcado el sistema; realizar generalización de los actores permite a los analistas facilitar el estudio de los requerimientos del sistema ya que esto permite agrupar las acciones a las que tengan acceso un conjunto determinado de usuarios de la aplicación.

Ilustración 6. Simbología de los Casos de Uso

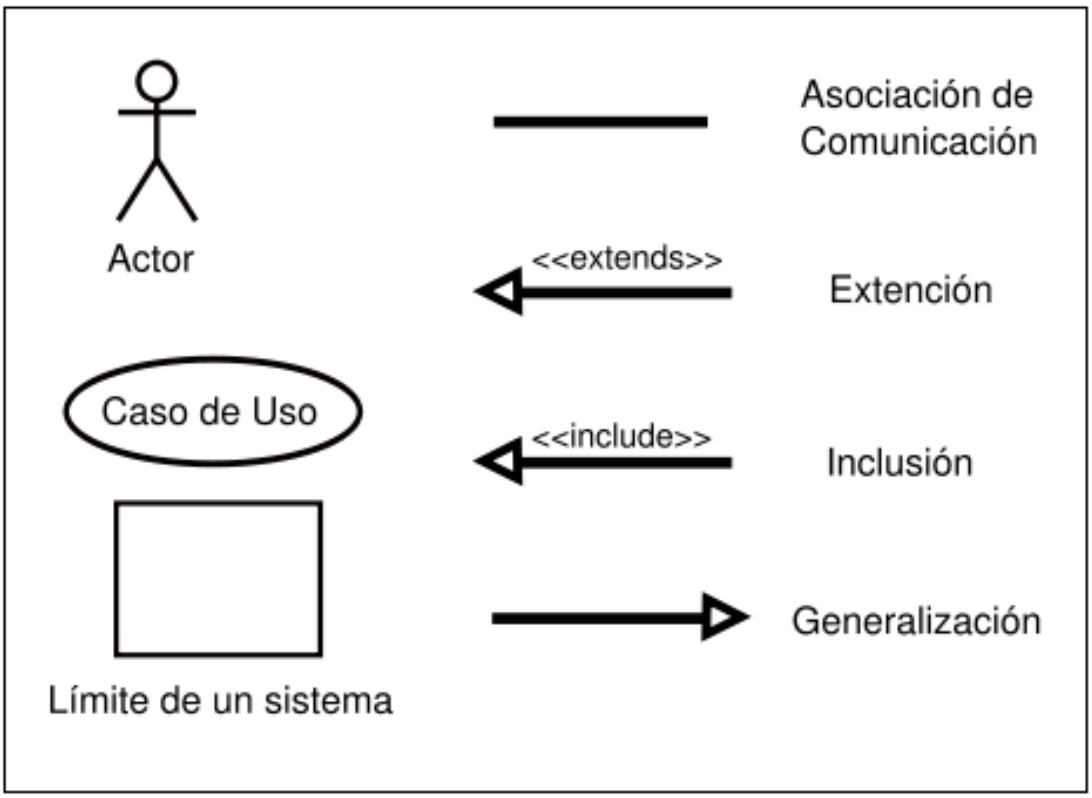
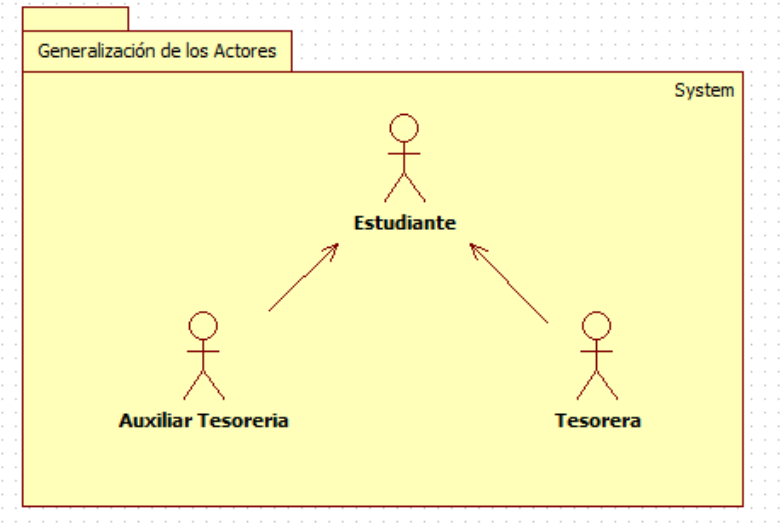
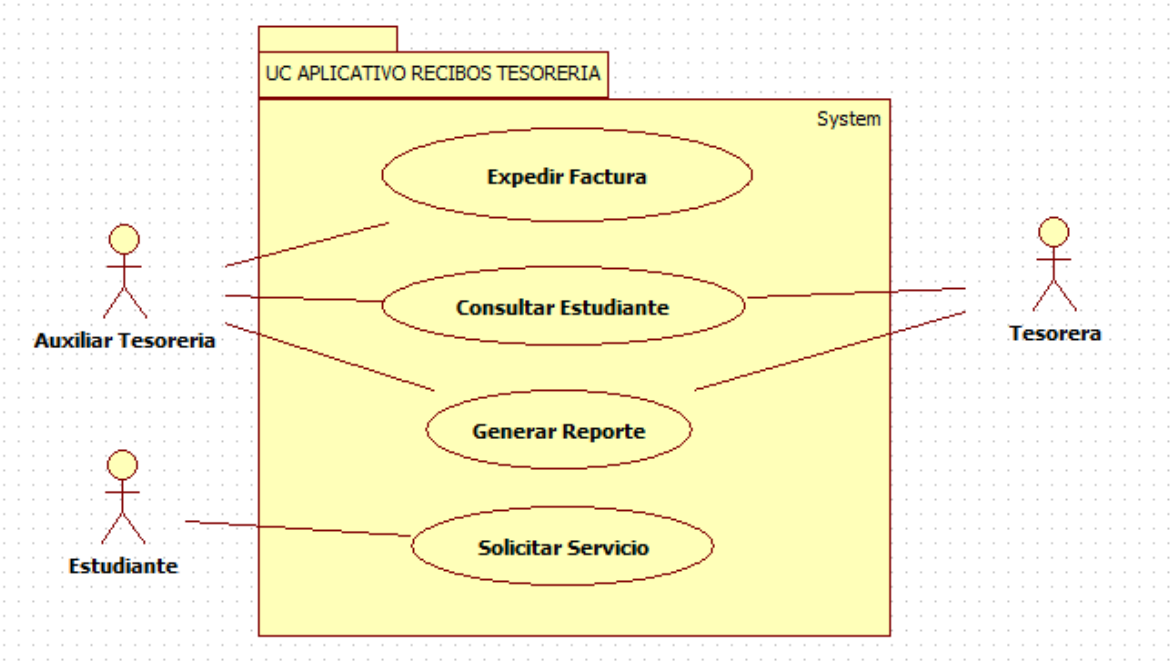


Ilustración 7. Generalización de los Actores



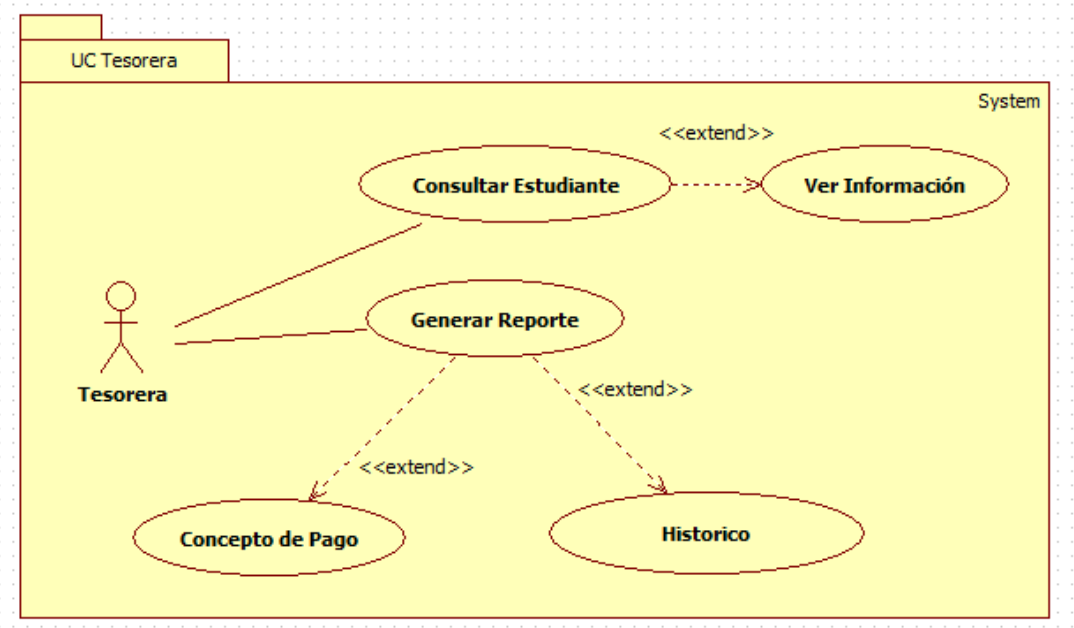
Fuente: Los autores

Ilustración 8. Diagrama de Caso de Uso. Programa de Recibos Tesorería



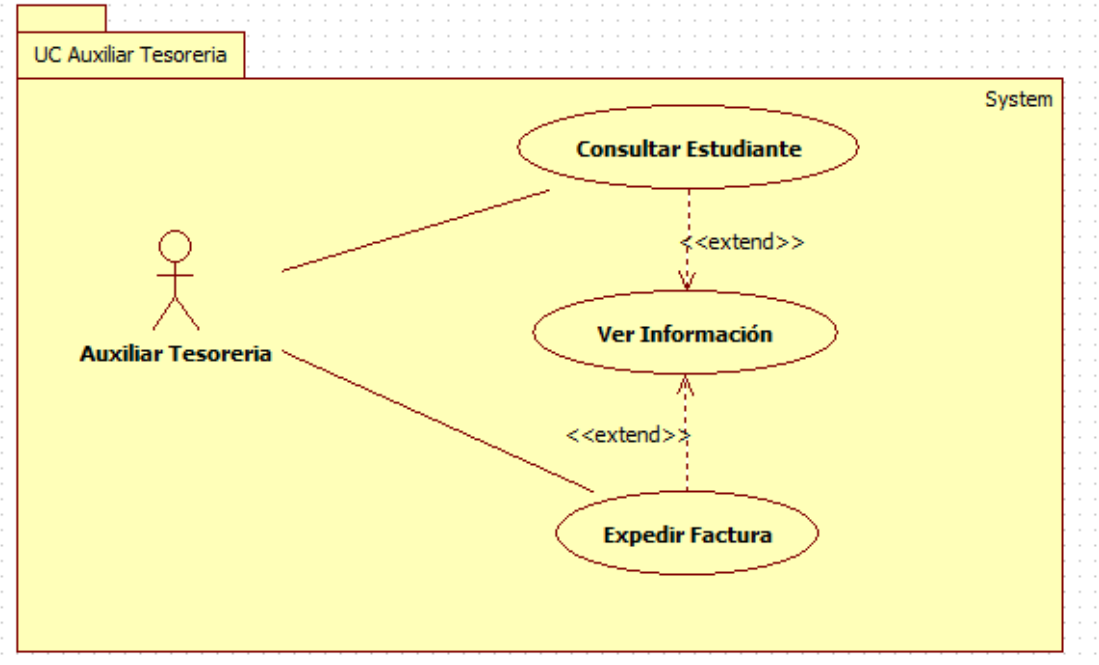
Fuente: Los autores

Ilustración 9. Diagrama de Caso de Uso. Tesorera



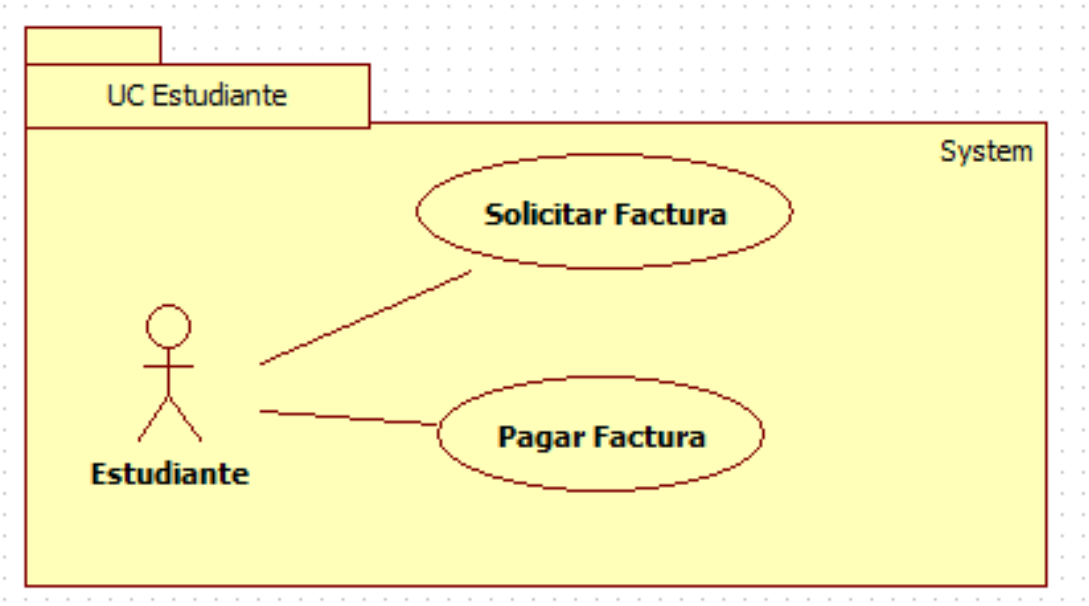
Fuente: Los autores

Ilustración 10. Diagrama de Caso de Uso. Auxiliar Tesorería



Fuente: Los autores

Ilustración 11. Diagrama de Caso de Uso. Estudiante



Fuente: Los autores

11.3 ANÁLISIS DE CASOS DE USO

11.3.1 Especificación de los Casos de Uso.

11.3.1.1 Especificación del Caso de Uso 1. Expedir Factura

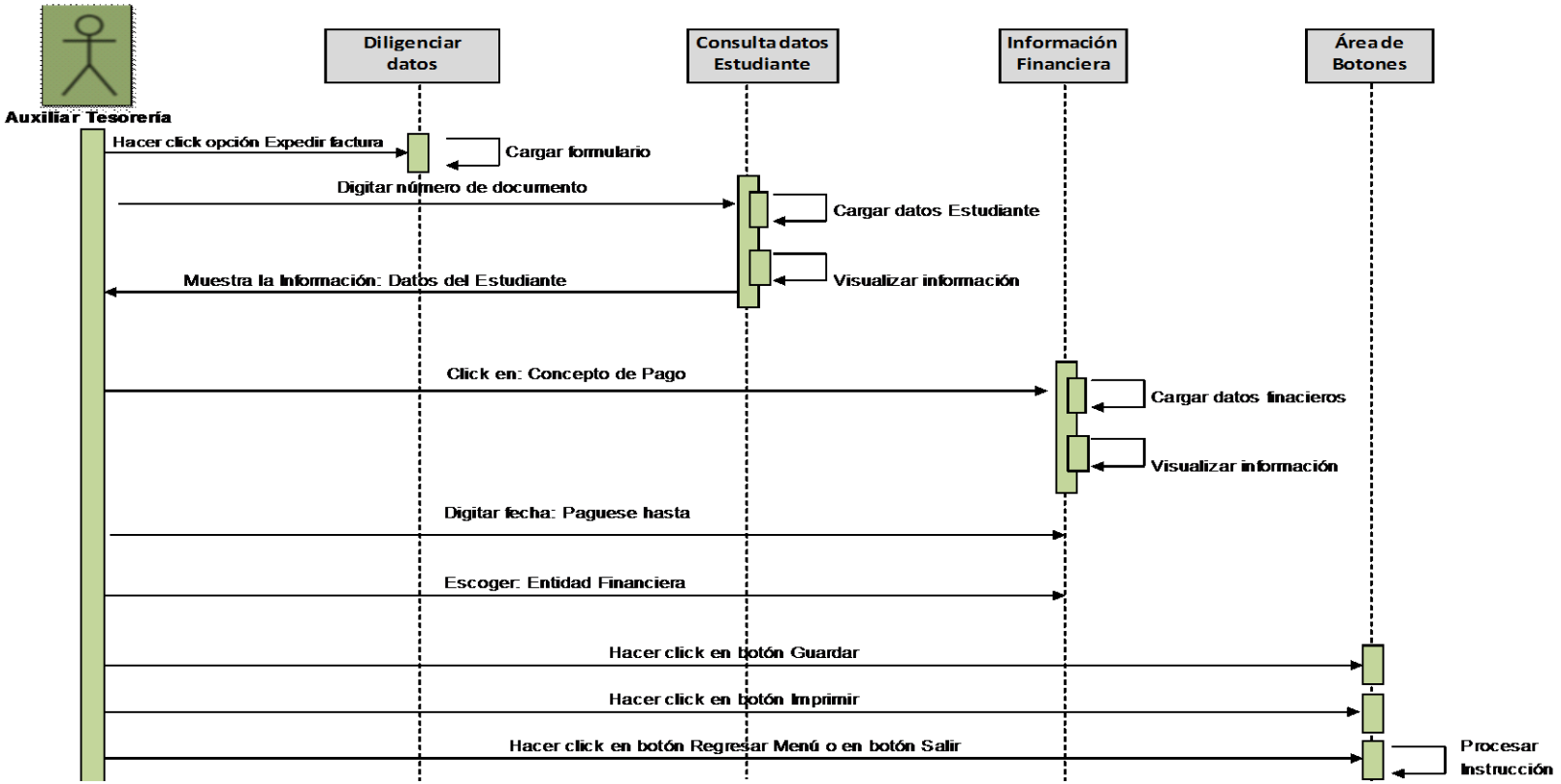
Tabla 9. Especificación del Caso de Uso 1. Expedir Factura

Nombre:	Expedir Factura
Autor:	Carlos A Huertas – José Julián Ruiz
Fecha:	15-Julio-2013
Descripción: Este evento permite al auxiliar de tesorería atender al estudiante quien solicita una factura de pago por el concepto que deba cancelar.	
Actores: Auxiliar de Tesorería – Estudiante	
Precondiciones: El auxiliar de tesorería debe recibir la solicitud de el (Los) estudiante(s) que deban realizar un pago por algún concepto en la universidad.	
Flujo Normal: El estudiante debe solicitar recibo de pago o factura a la auxiliar de tesorería en la ventanilla de la dependencia. La auxiliar toma los datos del estudiante como su documento de identidad para expedir la factura La auxiliar debe preguntar al estudiante el tipo de pago a realizar y digitar el concepto. El sistema genera recibo de pago de acuerdo al concepto que se ha solicitado y con el monto de dinero adeudado. El estudiante tiene la posibilidad de escoger en que banco va a cancelar. El estudiante tiene una fecha límite de pago, si la deja pasar debe ir nuevamente a tesorería para que le generen otro recibo. El(la) auxiliar de tesorería revisa el recibo diligenciado y si están correctos los datos se procede a guardar e imprimir. El estudiante recibe la factura y si su pago es en efectivo se debe desplazar a la entidad bancaria de su elección para realizar el pago. El sistema queda en el menú principal para seguir generando más recibos a los estudiantes; de lo contrario sale de la aplicación y termina.	
Flujo Alternativo: Se debe tener en cuenta que los conceptos son por multas de biblioteca y valores de menor cuantía que estime la Universidad Libre. El estudiante debe seleccionar el banco donde pagara el recibo: Bancolombia, Davivienda y de Occidente.	
Pos condiciones: Si el estudiante maneja tarjeta de crédito o debito puede cancelar directamente en tesorería.	

11.3.2 Diagrama de Secuencia del Caso de Uso.

11.3.2.1 Diagrama de Secuencia del Caso de Uso 1. Expedir Factura

Ilustración 12. Diagrama de Secuencia del Caso de Uso 1. Expedir Factura



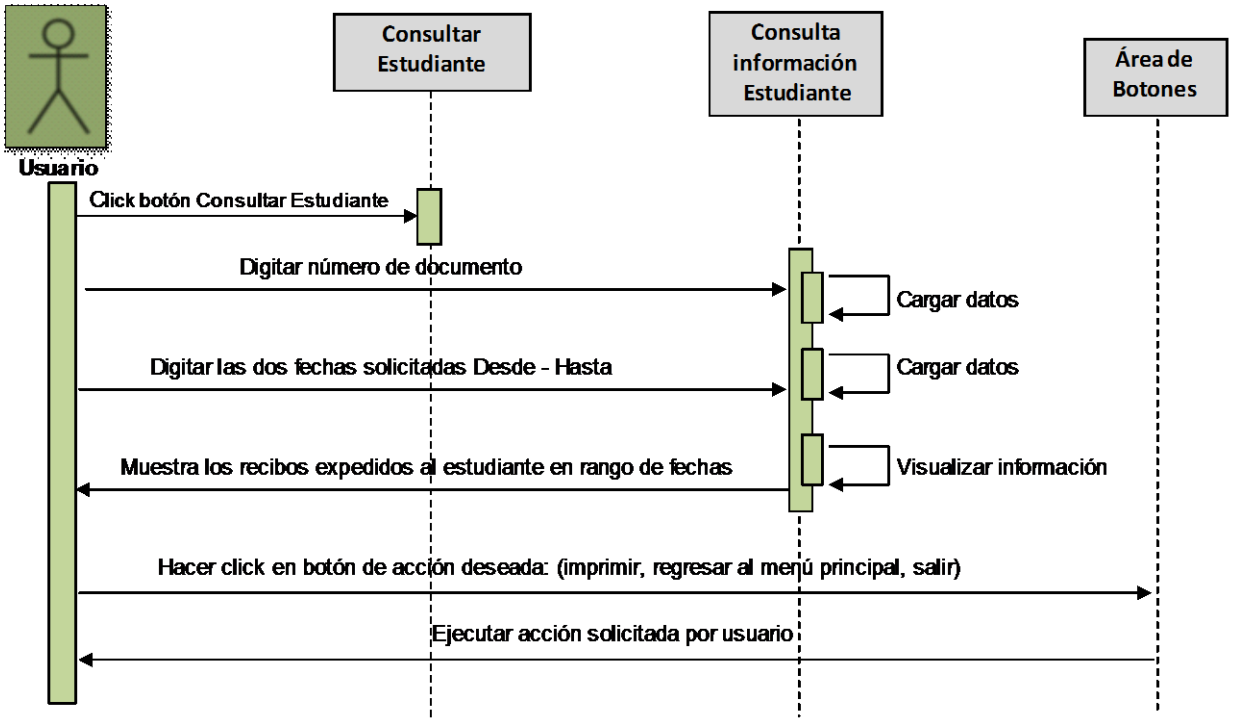
11.3.1.2 Especificación del Caso de Uso 2. Consultar Estudiante

Tabla 10. Especificación del Caso de Uso 2. Consultar Estudiante

Nombre:	Consultar Estudiante
Autor:	Carlos A Huertas – José Julián Ruiz
Fecha:	15-Julio-2013
Descripción: Permitir que el (la) auxiliar de tesorería y/o tesorera (o) a consideración y/o solicitud de un estudiante pueda consultar en un momento dado los recibos de pago que tenga pendientes o le hayan sido expedidos por cualquier concepto.	
Actores: Auxiliar Tesorería – Tesorera (o)	
Precondiciones: Consultas pendientes de un usuario del sistema. Solicitud de un estudiante	
Flujo Normal: El usuario del sistema requiere consultar en un momento específico los recibos expedidos a un estudiante El estudiante se deberá acercar a la tesorería y solicitar la consulta de los recibos pendientes por pagar o ya pagados. El Usuario del sistema deberá elegir la opción del menú consulta de estudiante El usuario del sistema deberá digitar la identificación del estudiante El usuario deberá digitar el rango de fechas de consulta El sistema genera un reporte de la cantidad de recibos que tiene pendientes y / o cancelados y sus respectivos valores y conceptos. El sistema de información se deberá regresar al menú principal a la espera de generar más recibos, si no es así, sale de la aplicación y termina.	
Flujo Alternativo: Ninguno.	
Pos condiciones: Si el estudiante no cancela los recibos pendientes no podrá acceder a libros en biblioteca, no podrá mirar notas y no podrá matricular materia, hasta que no se encuentre a paz y salvo.	

11.3.2.2 Diagrama de Secuencia del Caso de Uso 2. Consultar Estudiante

Ilustración 13. Diagrama de Secuencia del Caso de Uso 2. Consultar Estudiante



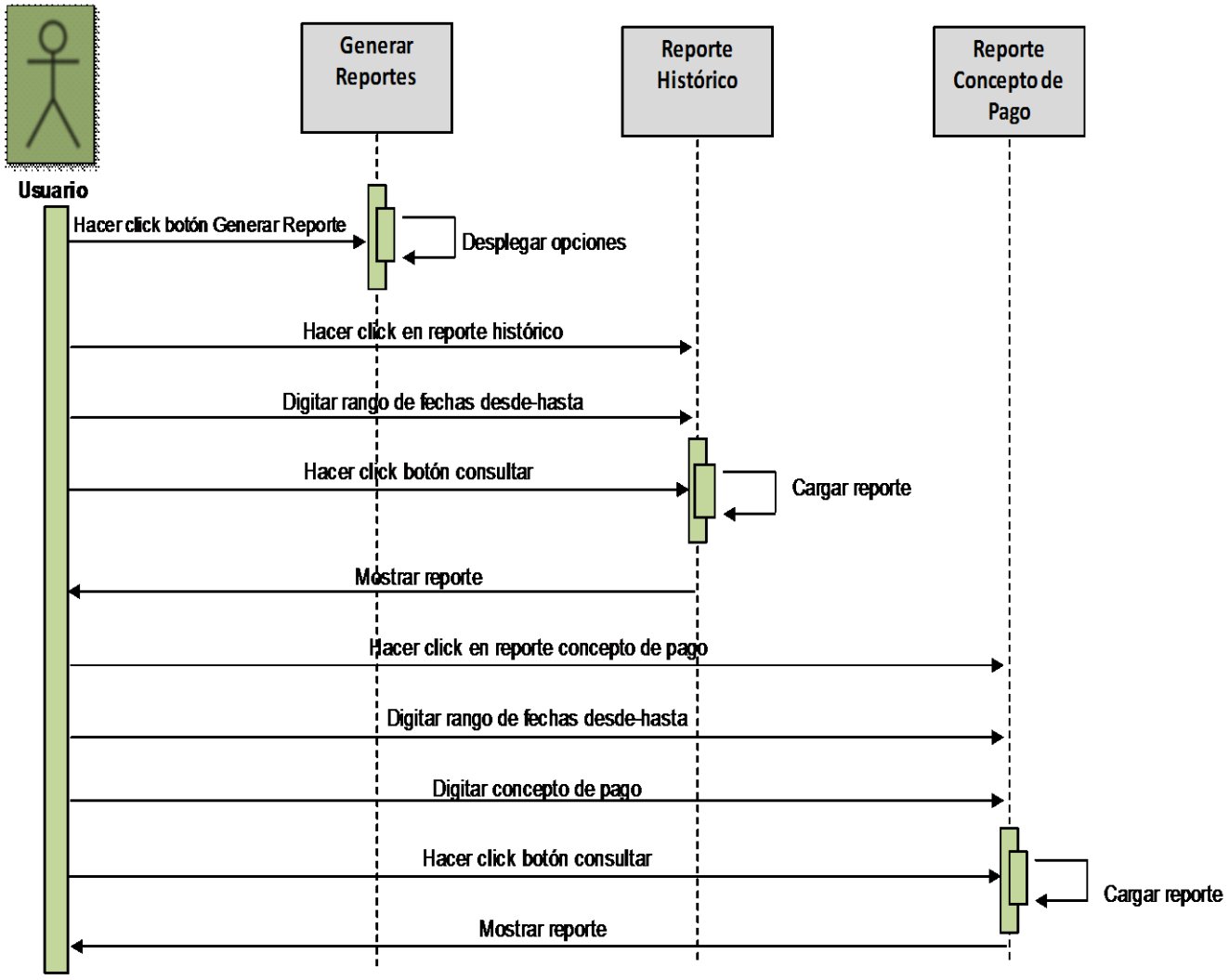
11.3.1.3 Especificación del Caso de Uso 3. Generar Reporte

Tabla 11. Especificación del Caso de Uso 3. Generar Reporte

Nombre:	Generar Reportes
Autor:	Carlos A Huertas – José Julián Ruiz
Fecha:	15-Julio-2013
Descripción: Permitir que el (la) Auxiliar de tesorería y/o el (la) Tesorero pueda consultar en cualquier momento los reportes generados en el sistema de información por histórico o por concepto de pago.	
Actores: Auxiliar Tesorería – Tesorera (o)	
Precondiciones: Que la base de datos haya sido previamente alimentada con información de facturas expedidas.	
Flujo Normal: El (la) Auxiliar de tesorería deberá elegir la opción reportes del menú principal y escoger el tipo de reporte que quiere visualizar y/o imprimir El sistema mostrará en la opción histórico, el rango de fechas que deberá digitar el usuario para que genere el reporte indicado de todos los recibos que han sido expedidos en el área. El sistema mostrará en la opción Concepto de pago, el rango de fechas y el concepto de pago que deberán ser elegidos por el usuario. El sistema generará una vez sea oprimido el botón consultar, el reporte indicado. El usuario debe elegir si desea imprimir el reporte o simplemente es de consulta en el sistema. El sistema de información queda a la espera de ser retornado al menú principal o salir de la aplicación y terminar.	
Flujo Alternativo: Ninguno.	
Pos condiciones: Ninguna.	

11.3.2.3 Diagrama de Secuencia del Caso de Uso 3. Generar Reporte

Ilustración 14. Diagrama de Secuencia del Caso de Uso 3. Generar Reporte



11.4 DIAGRAMA DE CLASES

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, orientada a objetos.

Es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre estos

El propósito de este diagrama es el de representar los objetos fundamentales del sistema, es decir los que percibe el usuario y con los que espera tratar para completar su tarea en vez de objetos del sistema o de un modelo de programación.

- La clase define el ámbito de definición de un conjunto de objetos.
- Cada objeto pertenece a una clase.
- Los objetos se crean por instanciación de las clases.

Ilustración 15. Diagrama de Clases Generalización

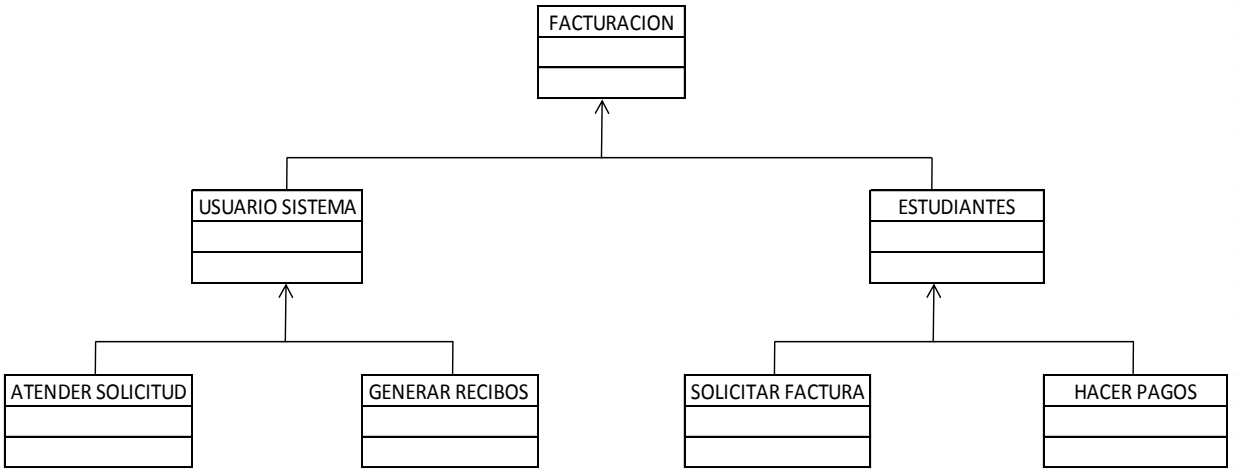
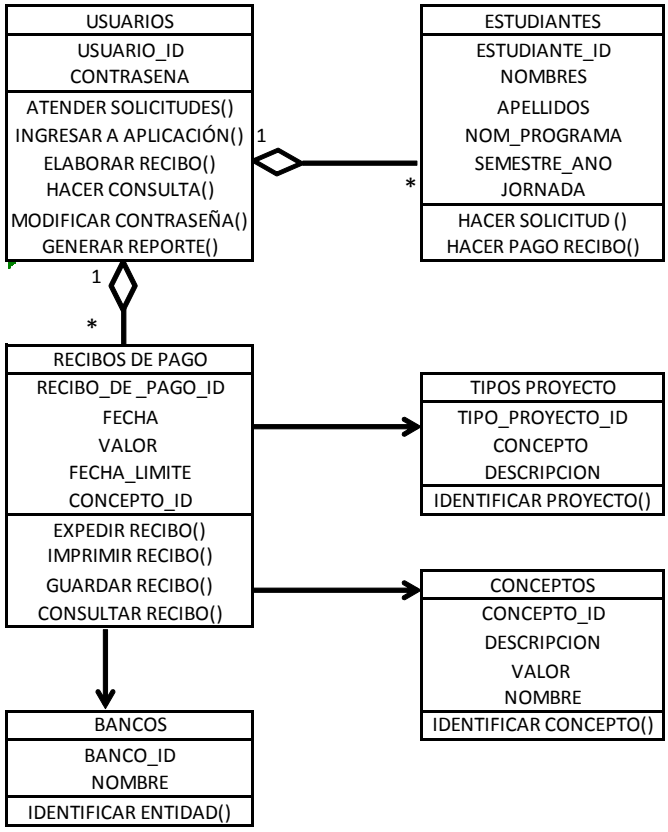


Ilustración 16. Diagrama de Clases Dependencia

Ilustración 17. Diagrama de Clases



12. DISEÑO DEL PROTOTIPO DE LA APLICACIÓN

12.1 DISEÑO DE LA BASE DE DATOS

La base de datos Oracle conocida como Oracle RDBMS o simplemente como Oracle, es un sistema de gestión de base de datos relacional de objetos producido y comercializado por Oracle Corporation.

Cuenta con la capacidad de almacenar y ejecutar procedimientos almacenados y funciones dentro de sí mismo. PL / SQL (extensión del procedimiento patentado por Oracle Corporation SQL) o el lenguaje orientado a objetos Java puede invocar tales objetos de código y / o proporcionar las estructuras de programación para escribirlos.

Con respecto a su almacenamiento, Los RDBMS Oracle almacena los datos lógicamente en la forma de tablas y físicamente en forma de datos de archivos (ficheros de datos) los espacios de tabla pueden contener diversos tipos de segmentos de memoria , tales como segmentos de datos, segmentos de indexación, etc y los segmentos a su vez comprender una o más extensiones . Las extensiones son grupos de bloques de datos contiguos. En los bloques de datos se forman las unidades básicas de almacenamiento de datos.

Una unidad puede imponer capacidad máxima de almacenamiento por usuario en cada espacio de tablas.

Las nuevas versiones de la base de datos también pueden incluir una partición de función lo que significa que permite la partición de tablas basadas en diferentes conjuntos de claves. Particiones específicas se pueden agregar o quitar para ayudar a manejar grandes conjuntos de datos con facilidad.

Sistemas Operativos que la acogen:

Oracle Corporation ha consolidado las siguientes plataformas de sistemas operativos.

- zLinux 64
- Microsoft Windows (32 bits)
- Microsoft Windows (x64)
- Linux x86
- Linux x86-64
- Solaris (SPARC)
- Solaris (x86-64)
- HP-UX Itanium
- HP-UX PA-RISC (64-bit)
- AIX (PPC64)
- OpenVMS (IA64)

Características de la Base de Datos Oracle

Historia Sesiones activas (ASH), que recoge datos para la vigilancia inmediata de la actividad de base de datos muy reciente.

Repositorio de Carga de Trabajo Automática (AWR) , la prestación de servicios de vigilancia a las instalaciones de la base de datos de Oracle desde la versión Oracle 10. Antes de la versión de Oracle versión 10, la instalación Statspack proporcionado una funcionalidad similar.

Clusterware

La agregación de datos y consolidación

Data Guard para alta disponibilidad

Conectividad genérica para la conexión a los sistemas que no son de Oracle.

Utilidades de la bomba de datos, que ayuda en la importación y exportación de datos y metadatos entre bases de datos.

Gestor de Base de Datos de Recursos (DRM), que controla el uso de los recursos informáticos.

Rollback paralelo Fast-start.

Auditoría de grano fino (FGA) (en Oracle Enterprise Edition). Suplementos funciones estándar de auditoría de seguridad.

Flashback para la recuperación de datos selectivos y la reconstrucción

iSQLPlus , un navegador basado en interfaz gráfica de usuario (GUI) para la base de datos Oracle de manipulación de datos (comparación de SQL * Plus)

Oracle Data Access Components (ODAC), herramientas que consisten en:

- Oracle Data Provider para. NET (ODP.NET) [92]
- Oracle Developer Tools (ODT) para Visual Studio
- Proveedores de Oracle para ASP.NET
- Oracle Database Extensions para. NET
- Proveedor Oracle para OLE DB
- Objetos de Oracle para OLE
- Oracle Services para Microsoft Transaction Server

Archivos Oracle gestionados (OMF). Una característica que permite nombramiento automatizado, la creación y supresión de ficheros de datos a nivel de sistema operativo.

Recovery Manager (RMAN) para la base de datos de: Copia de seguridad, restauración y recuperación.

SQL * Plus , un programa que permite a los usuarios interactuar con la base de datos Oracle a través de SQL y PL / SQL comandos en una línea de comandos.

Conexión universal libre (UCP), una agrupación de conexiones basadas en Java y soporte JDBC , LDAP, y JCA

Virtual Private Database.(VPD), una aplicación de control de acceso de grano fino.

Los usuarios pueden desarrollar sus propias aplicaciones en Java y PL / SQL utilizando herramientas tales como:

- Oracle Forms
- Oracle JDeveloper
- Oracle Reports

A partir de 2007 Oracle Corporation había comenzado una unidad hacia el "wizard" impulsadas por entornos con el fin de que los no programadores producir simples aplicaciones de base de datos.

JAccelerator (NCOMP) un nativo de la compilación de Java "acelerador", se integra con optimización de hardware de código Java en una base de datos Oracle.

Oracle SQL Developer , una herramienta gráfica libre para el desarrollo de bases de datos, permite a los desarrolladores navegar por los objetos de base de datos, ejecutar sentencias SQL y scripts de SQL y editar y depurar PL / SQL. Incorporación de informes estándar y personalizados.

El SQLTXPLAIN herramienta (SQLT) proporciona asistencia de ajuste para Oracle SQL consultas.

Para el prototipo de la aplicación a implementar se realiza el diseño de una base de datos relacional que tiene como fin permitir almacenar en la tabla Estudiantes, la información concerniente a este, en la tabla Recibos de pago cada uno de los recibos que le han sido expedidos por los diferentes conceptos, la tabla Conceptos cada uno de los ítems por los que se elabora un recibo de pago con sus datos, la tabla Proyectos donde se determina si el concepto de pago es de tipo Operacional y No operacional, la tabla Banco con la información de cada entidad y finalmente la tabla Usuarios, para la creación de los usuarios que tendrán acceso al sistema.

Ilustración 18. Modelo Relacional

Clases.

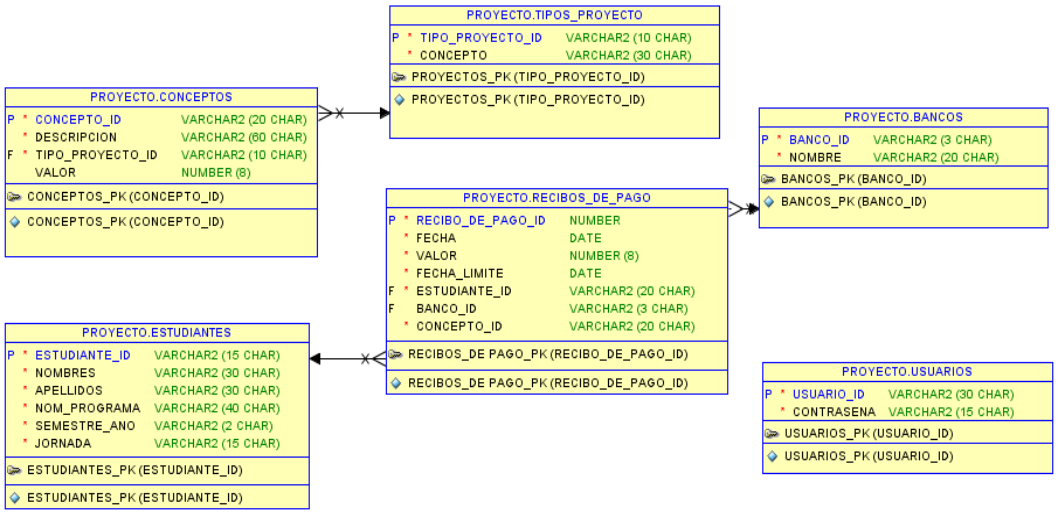


Ilustración 19. Modelo Lógico

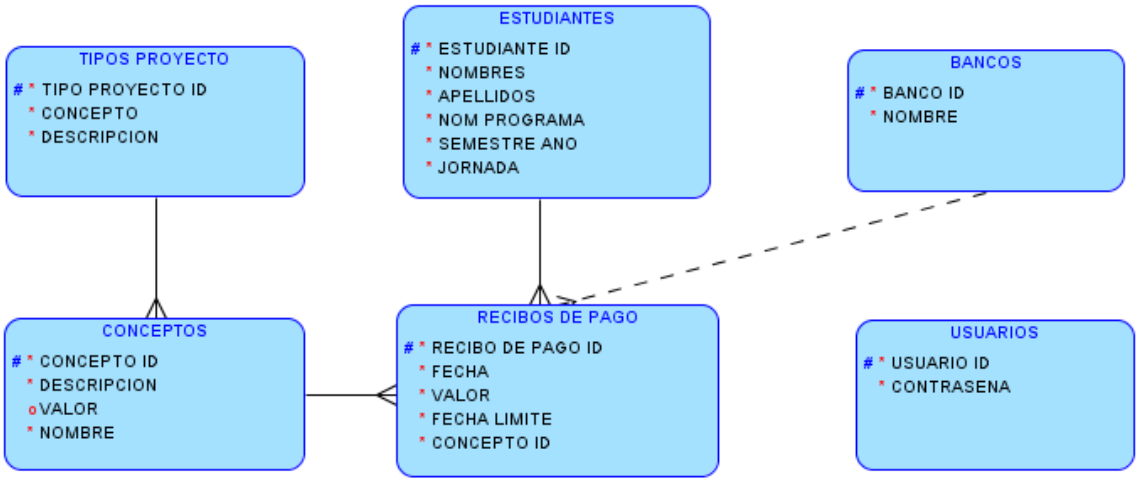


Tabla 12. Tabla Estudiantes

PROYECTO.ESTUDIANTES		
P *	ESTUDIANTE_ID	VARCHAR2 (15 CHAR)
*	NOMBRES	VARCHAR2 (30 CHAR)
*	APELLIDOS	VARCHAR2 (30 CHAR)
*	NOM_PROGRAMA	VARCHAR2 (40 CHAR)
*	SEMESTRE_ANO	VARCHAR2 (2 CHAR)
*	JORNADA	VARCHAR2 (15 CHAR)
ESTUDIANTES_PK (ESTUDIANTE_ID)		
ESTUDIANTES_PK (ESTUDIANTE_ID)		

Tabla 13. Tabla Recibos_de pago

PROYECTO.RECIBOS_DE_PAGO		
P *	RECIBO_DE_PAGO_ID	NUMBER
*	FECHA	DATE
*	VALOR	NUMBER (8)
*	FECHA_LIMITE	DATE
F *	ESTUDIANTE_ID	VARCHAR2 (20 CHAR)
F	BANCO_ID	VARCHAR2 (3 CHAR)
*	CONCEPTO_ID	VARCHAR2 (20 CHAR)
RECIBOS_DE PAGO_PK (RECIBO_DE_PAGO_ID)		
RECIBOS_DE PAGO_PK (RECIBO_DE_PAGO_ID)		

Tabla 14. Tabla Conceptos

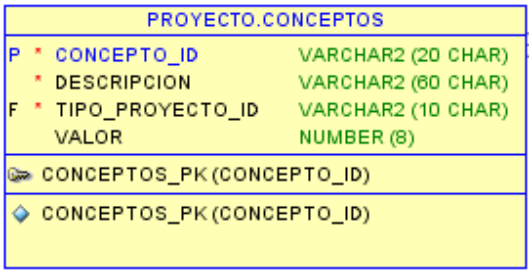


Tabla 15. Tabla Bancos

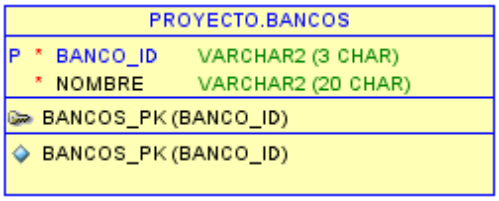


Tabla 16. Tabla Proyectos

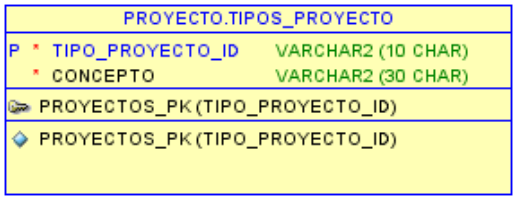
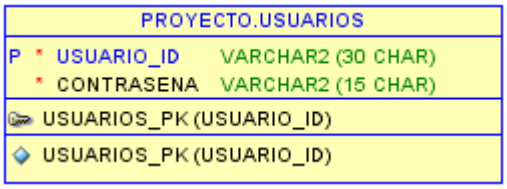


Tabla 17. Tabla Usuarios



12.2 CÓDIGO ORACLE PARA CREAR TABLAS EN LA BASE DE DATOS

12.2.1 Código Tabla Estudiantes

```
--------------------------------------------------------
-- Archivo creado  - miércoles-agosto-28-2013
--------------------------------------------------------
--------------------------------------------------------
--  DDL for Table ESTUDIANTES
--------------------------------------------------------

  CREATE TABLE "PROYECTO"."ESTUDIANTES"
   (	"ESTUDIANTE_ID" VARCHAR2(15 CHAR),
	"NOMBRES" VARCHAR2(30 CHAR),
	"APELLIDOS" VARCHAR2(30 CHAR),
	"NOM_PROGRAMA" VARCHAR2(40 CHAR),
	"SEMESTRE_ANO" VARCHAR2(2 CHAR),
	"JORNADA" VARCHAR2(15 CHAR)
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT
FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;

   COMMENT ON COLUMN "PROYECTO"."ESTUDIANTES"."ESTUDIANTE_ID" IS
'Corresponde al número de documento de identidad de cada estudiante ';
   COMMENT ON COLUMN "PROYECTO"."ESTUDIANTES"."NOMBRES" IS
'Corresponde al nombre(s) del estudiante';
   COMMENT ON COLUMN "PROYECTO"."ESTUDIANTES"."APELLIDOS" IS
'Corresponde al apellido(s) del estudiante';
   COMMENT ON COLUMN "PROYECTO"."ESTUDIANTES"."NOM_PROGRAMA" IS
'Hace referencia al nombre del programa en que se encuentra matriculado el estudiante';
   COMMENT ON COLUMN "PROYECTO"."ESTUDIANTES"."SEMESTRE_ANO" IS
'Hace referencia al semestre o año que cursa el estudiante';
   COMMENT ON COLUMN "PROYECTO"."ESTUDIANTES"."JORNADA" IS 'Hace
referencia a la jornada en que se encuentra matriculado el estudiante';
--------------------------------------------------------
--  DDL for Index ESTUDIANTES_PK
--------------------------------------------------------

  CREATE UNIQUE INDEX "PROYECTO"."ESTUDIANTES_PK" ON
"PROYECTO"."ESTUDIANTES" ("ESTUDIANTE_ID")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
```

```
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT
FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;
--------------------------------------------------------
--  Constraints for Table ESTUDIANTES
--------------------------------------------------------

  ALTER TABLE "PROYECTO"."ESTUDIANTES" ADD CONSTRAINT
"ESTUDIANTES_PK" PRIMARY KEY ("ESTUDIANTE_ID")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT
FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS"  ENABLE;
  ALTER TABLE "PROYECTO"."ESTUDIANTES" MODIFY ("JORNADA" NOT NULL
ENABLE);
  ALTER TABLE "PROYECTO"."ESTUDIANTES" MODIFY ("SEMESTRE_ANO" NOT
NULL ENABLE);
  ALTER TABLE "PROYECTO"."ESTUDIANTES" MODIFY ("NOM_PROGRAMA" NOT
NULL ENABLE);
  ALTER TABLE "PROYECTO"."ESTUDIANTES" MODIFY ("APELLIDOS" NOT NULL
ENABLE);
  ALTER TABLE "PROYECTO"."ESTUDIANTES" MODIFY ("NOMBRES" NOT NULL
ENABLE);
  ALTER TABLE "PROYECTO"."ESTUDIANTES" MODIFY ("ESTUDIANTE_ID" NOT
NULL ENABLE);
```

12.2.2 Código Tabla Recibos de Pago

```
--------------------------------------------------------
-- Archivo creado  - miércoles-agosto-28-2013
--------------------------------------------------------
--------------------------------------------------------
--  DDL for Table RECIBOS_DE_PAGO
--------------------------------------------------------

  CREATE TABLE "PROYECTO"."RECIBOS_DE_PAGO"
   (	"RECIBO_DE_PAGO_ID" NUMBER(*,0),
	"FECHA" DATE,
	"VALOR" NUMBER(8,0),
	"FECHA_LIMITE" DATE,
	"ESTUDIANTE_ID" VARCHAR2(15 CHAR),
	"BANCO_ID" VARCHAR2(3 CHAR),
	"CONCEPTO_ID" VARCHAR2(20 CHAR)
```

```
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT
FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;

   COMMENT ON COLUMN
"PROYECTO"."RECIBOS_DE_PAGO"."RECIBO_DE_PAGO_ID" IS 'Corresponde al
consecutivo del recibo de pago';
   COMMENT ON COLUMN "PROYECTO"."RECIBOS_DE_PAGO"."FECHA" IS 'Indica la
fecha en que se elabora el recibo de pago';
   COMMENT ON COLUMN "PROYECTO"."RECIBOS_DE_PAGO"."VALOR" IS
'Corresponde al monto que debe pagar el estudiante por el concepto generado';
   COMMENT ON COLUMN "PROYECTO"."RECIBOS_DE_PAGO"."FECHA_LIMITE" IS
'Indica la fecha máxima en que puede cancelar el recibo de pago';
   COMMENT ON COLUMN "PROYECTO"."RECIBOS_DE_PAGO"."ESTUDIANTE_ID" IS
'Corresponde al documento de identidad de cada estudiante';
   COMMENT ON COLUMN "PROYECTO"."RECIBOS_DE_PAGO"."BANCO_ID" IS
'Corresponde al número que identifica la entidad bancaria';
   COMMENT ON COLUMN "PROYECTO"."RECIBOS_DE_PAGO"."CONCEPTO_ID" IS
'Corresponde al número que identifica al concepto de pago';
--------------------------------------------------------
--  DDL for Index RECIBOS_DE PAGO_PK
--------------------------------------------------------

  CREATE UNIQUE INDEX "PROYECTO"."RECIBOS_DE PAGO_PK" ON
"PROYECTO"."RECIBOS_DE_PAGO" ("RECIBO_DE_PAGO_ID")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT
FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;
--------------------------------------------------------
--  Constraints for Table RECIBOS_DE_PAGO
--------------------------------------------------------

  ALTER TABLE "PROYECTO"."RECIBOS_DE_PAGO" MODIFY ("ESTUDIANTE_ID"
NOT NULL ENABLE);
  ALTER TABLE "PROYECTO"."RECIBOS_DE_PAGO" ADD CONSTRAINT
"RECIBOS_DE PAGO_PK" PRIMARY KEY ("RECIBO_DE_PAGO_ID")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT
FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS"  ENABLE;
  ALTER TABLE "PROYECTO"."RECIBOS_DE_PAGO" MODIFY ("FECHA_LIMITE" NOT
NULL ENABLE);
```

```
  ALTER TABLE "PROYECTO"."RECIBOS_DE_PAGO" MODIFY ("VALOR" NOT NULL
ENABLE);
  ALTER TABLE "PROYECTO"."RECIBOS_DE_PAGO" MODIFY ("FECHA" NOT NULL
ENABLE);
  ALTER TABLE "PROYECTO"."RECIBOS_DE_PAGO" MODIFY
("RECIBO_DE_PAGO_ID" NOT NULL ENABLE);
  ALTER TABLE "PROYECTO"."RECIBOS_DE_PAGO" MODIFY ("CONCEPTO_ID" NOT
NULL ENABLE);
--------------------------------------------------------
--  Ref Constraints for Table RECIBOS_DE_PAGO
--------------------------------------------------------

  ALTER TABLE "PROYECTO"."RECIBOS_DE_PAGO" ADD CONSTRAINT
"RECIBOS_DE_PAGO_BANCOS_FK1" FOREIGN KEY ("BANCO_ID")
              REFERENCES "PROYECTO"."BANCOS" ("BANCO_ID") ON DELETE
CASCADE ENABLE;
  ALTER TABLE "PROYECTO"."RECIBOS_DE_PAGO" ADD CONSTRAINT
"RECIBOS_DE_PAGO_CONCEPTO_FK" FOREIGN KEY ("ESTUDIANTE_ID")
              REFERENCES "PROYECTO"."CONCEPTOS" ("CONCEPTO_ID") ON
DELETE CASCADE ENABLE;
  ALTER TABLE "PROYECTO"."RECIBOS_DE_PAGO" ADD CONSTRAINT
"RECIBOS_DE_PAGO_ESTUD_FK1" FOREIGN KEY ("ESTUDIANTE_ID")
              REFERENCES "PROYECTO"."ESTUDIANTES" ("ESTUDIANTE_ID") ON
DELETE CASCADE ENABLE;
--------------------------------------------------------
--  DDL for Trigger INCRE_ID_RECIBO
--------------------------------------------------------

  CREATE OR REPLACE TRIGGER "PROYECTO"."INCRE_ID_RECIBO"
   before insert on "PROYECTO"."RECIBOS_DE_PAGO"
   for each row
begin
   if inserting then
      if :NEW."RECIBO_DE_PAGO_ID" is null then
         select SEQ_ID_RECIBO.nextval into :NEW."RECIBO_DE_PAGO_ID" from dual;
      end if;
   end if;
end;

/
ALTER TRIGGER "PROYECTO"."INCRE_ID_RECIBO" ENABLE;
```

12.2.3 Código Tabla Conceptos

```
--------------------------------------------------------
-- Archivo creado  - miércoles-agosto-28-2013
--------------------------------------------------------
--------------------------------------------------------
--  DDL for Table CONCEPTOS
--------------------------------------------------------

  CREATE TABLE "PROYECTO"."CONCEPTOS"
   (	"CONCEPTO_ID" VARCHAR2(20 CHAR),
	"DESCRIPCION" VARCHAR2(60 CHAR),
	"TIPO_PROYECTO_ID" VARCHAR2(10 CHAR),
	"VALOR" NUMBER(8,0)
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT
FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;

   COMMENT ON COLUMN "PROYECTO"."CONCEPTOS"."CONCEPTO_ID" IS
'Corresponde al número de concepto de pago';
   COMMENT ON COLUMN "PROYECTO"."CONCEPTOS"."DESCRIPCION" IS
'Corresponde al nombre del concepto de pago';
   COMMENT ON COLUMN "PROYECTO"."CONCEPTOS"."TIPO_PROYECTO_ID" IS
'Corresponde al número que identifica al proyecto';
   COMMENT ON COLUMN "PROYECTO"."CONCEPTOS"."VALOR" IS 'Es el monto que
corresponde al pago que debe hacer el estudiante';
--------------------------------------------------------
--  DDL for Index CONCEPTOS_PK
--------------------------------------------------------

  CREATE UNIQUE INDEX "PROYECTO"."CONCEPTOS_PK" ON
"PROYECTO"."CONCEPTOS" ("CONCEPTO_ID")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT
FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;
--------------------------------------------------------
--  Constraints for Table CONCEPTOS
--------------------------------------------------------

  ALTER TABLE "PROYECTO"."CONCEPTOS" MODIFY ("TIPO_PROYECTO_ID" NOT
NULL ENABLE);
```

```
  ALTER TABLE "PROYECTO"."CONCEPTOS" ADD CONSTRAINT "CONCEPTOS_PK"
PRIMARY KEY ("CONCEPTO_ID")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT
FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS"  ENABLE;
  ALTER TABLE "PROYECTO"."CONCEPTOS" MODIFY ("DESCRIPCION" NOT NULL
ENABLE);
  ALTER TABLE "PROYECTO"."CONCEPTOS" MODIFY ("CONCEPTO_ID" NOT NULL
ENABLE);
--------------------------------------------------------
--  Ref Constraints for Table CONCEPTOS
--------------------------------------------------------

  ALTER TABLE "PROYECTO"."CONCEPTOS" ADD CONSTRAINT
"CONCEPTOS_TIPOS_PROYECTO_FK1" FOREIGN KEY ("TIPO_PROYECTO_ID")
              REFERENCES "PROYECTO"."TIPOS_PROYECTO" ("TIPO_PROYECTO_ID")
ON DELETE CASCADE ENABLE;
```

12.2.4 Código Tabla Bancos

```
--------------------------------------------------------
-- Archivo creado  - miércoles-agosto-28-2013
--------------------------------------------------------
--------------------------------------------------------
--  DDL for Table BANCOS
--------------------------------------------------------

  CREATE TABLE "PROYECTO"."BANCOS"
   (          "BANCO_ID" VARCHAR2(3 CHAR),
              "NOMBRE" VARCHAR2(20 CHAR)
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT
FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;

   COMMENT ON COLUMN "PROYECTO"."BANCOS"."BANCO_ID" IS 'Es el código de
identificación de la entidad bancaria';
   COMMENT ON COLUMN "PROYECTO"."BANCOS"."NOMBRE" IS 'Es el nombre de la
entidad financiera';
```

12.2.5 Código Tabla Tipo Proyecto

```
--------------------------------------------------------
-- Archivo creado  - miércoles-agosto-28-2013
--------------------------------------------------------
--------------------------------------------------------
--  DDL for Table TIPOS_PROYECTO
--------------------------------------------------------

  CREATE TABLE "PROYECTO"."TIPOS_PROYECTO"
   (	"TIPO_PROYECTO_ID" VARCHAR2(10 CHAR),
	"CONCEPTO" VARCHAR2(30 CHAR)
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT
FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;

   COMMENT ON COLUMN
"PROYECTO"."TIPOS_PROYECTO"."TIPO_PROYECTO_ID" IS 'Corresponde al número
que identifica al proyecto';
   COMMENT ON COLUMN "PROYECTO"."TIPOS_PROYECTO"."CONCEPTO" IS 'Hace
referencia al nombre que identifica al proyecto';
--------------------------------------------------------
--  DDL for Index PROYECTOS_PK
--------------------------------------------------------

  CREATE UNIQUE INDEX "PROYECTO"."PROYECTOS_PK" ON
"PROYECTO"."TIPOS_PROYECTO" ("TIPO_PROYECTO_ID")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT
FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;
--------------------------------------------------------
--  Constraints for Table TIPOS_PROYECTO
--------------------------------------------------------

  ALTER TABLE "PROYECTO"."TIPOS_PROYECTO" ADD CONSTRAINT
"PROYECTOS_PK" PRIMARY KEY ("TIPO_PROYECTO_ID")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT
FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS"  ENABLE;
```

```
  ALTER TABLE "PROYECTO"."TIPOS_PROYECTO" MODIFY ("CONCEPTO" NOT
NULL ENABLE);
  ALTER TABLE "PROYECTO"."TIPOS_PROYECTO" MODIFY ("TIPO_PROYECTO_ID"
NOT NULL ENABLE);
```

12.2.6 Código Tabla Usuarios

```
--------------------------------------------------------
-- Archivo creado  - miércoles-agosto-28-2013
--------------------------------------------------------
--------------------------------------------------------
--  DDL for Table USUARIOS
--------------------------------------------------------

  CREATE TABLE "PROYECTO"."USUARIOS"
   (	"USUARIO_ID" VARCHAR2(30 CHAR),
	"CONTRASENA" VARCHAR2(15 CHAR)
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT
FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;

   COMMENT ON COLUMN "PROYECTO"."USUARIOS"."USUARIO_ID" IS 'Corresponde
al nombre del usuario creado para manipular la aplicación';
   COMMENT ON COLUMN "PROYECTO"."USUARIOS"."CONTRASENA" IS 'Hace
referencia a un código secreto que determina el usuario para hacer uso de la aplicación';
--------------------------------------------------------
--  DDL for Index USUARIOS_PK
--------------------------------------------------------

  CREATE UNIQUE INDEX "PROYECTO"."USUARIOS_PK" ON
"PROYECTO"."USUARIOS" ("USUARIO_ID")
  PCTFREE 10 INITRANS 2 MAXTRANS 255
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT
FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;
--------------------------------------------------------
--  Constraints for Table USUARIOS
--------------------------------------------------------
```

```
  ALTER TABLE "PROYECTO"."USUARIOS" ADD CONSTRAINT "USUARIOS_PK"
PRIMARY KEY ("USUARIO_ID")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT
FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS"  ENABLE;
  ALTER TABLE "PROYECTO"."USUARIOS" MODIFY ("CONTRASENA" NOT NULL
ENABLE);
  ALTER TABLE "PROYECTO"."USUARIOS" MODIFY ("USUARIO_ID" NOT NULL
ENABLE);
```

13. MARCO METODOLÓGICO

A. Tipo de investigación

El tipo de investigación que se va a realizar será cuantitativa ya que se conoce claramente el problema, se tienen objetivos delimitados y se ha planteado una hipótesis que permite definir un camino para llegar a la solución.

Se ha utilizado como instrumento de medición la entrevista estructurada, que es básicamente una conversación que se ha tenido con las funcionarias del área de tesorería y que en investigación cuantitativa tiene la particularidad fundamental que a todas las personas entrevistadas se les pregunta lo mismo y de la misma forma, y las respuestas que se obtienen de las preguntas son registradas también de la misma forma.

B. Método de Investigación

De acuerdo con la situación planteada en esta investigación y como se señala en los objetivos propuestos, los datos necesarios para el estudio se recolectaron de forma directa a través del personal que labora en el área de tesorería de la universidad en ambas sedes, por lo que el trabajo se enmarco en un diseño de campo.

C. Técnicas para la Recolección de la Información

Las técnicas para recolección de datos son todos aquellos recursos utilizados para la obtención de información, la cual permite interactuar con la problemática y poder extraer así, toda la información necesaria para lograr el desarrollo de la investigación. En el siguiente trabajo de investigación se llevaron a cabo diferentes técnicas para la recolección de los datos necesarios, estas son:

Observación directa Se utilizó la observación directa a través de la cual se pudo identificar detalladamente el procedimiento que realiza el área de tesorería desde el momento en que recibe una solicitud de servicio por parte de los usuarios, hasta que finaliza todas las gestiones necesarias para hacer entrega de dicho recibo y así poder establecer los requerimientos funcionales y no funcionales.

La Entrevista: Como se observóen el capítulo 11 del Análisis de Requerimientos, se diseñó un formato para entrevistar tanto a la Tesorera, como a la Auxiliar de tesorería; se utilizó este método igualmente como instrumento de recolección de datos, para conocer un poco más a fondo la problemática y las necesidades y poder determinar los requerimientos que deberá tener la nueva aplicación. Puede entonces definirse como la relación que se establece entre el investigador y los sujetos de estudio.

Procedimiento de recolección de datos La elaboración de la entrevista se realizó de forma manual utilizando una libreta de anotaciones para puntualizar los datos importantes y más relevantes de las necesidades con que se cuenta. Se contempló una serie de preguntas con la finalidad de obtener respuestas libres que proporcionaron la mayor cantidad de información posible con el objeto de afianzar y confirmar ideas sobre el diseño de la aplicación y de la base de datos, que ya se pudo contemplar en el anterior capítulo.

Aplicación de la entrevista Para la aplicación de la entrevista se eligieron personas claves que forman parte del área de tesorería y que son los involucrados directamente en las gestiones relacionadas con la prestación de servicios.

Observación de la aplicación actual Por medio de la observación se realizó un estudio del entorno de operación actual y las gestiones que realiza el área de tesorería antes de prestar un servicio, analizando todo lo que se pudo percibir con la intención de comprender lo que realmente sucede y a lo que se quiere llegar con la implementación de la base de datos.

Recopilación y análisis El proceso de consulta y recolección de datos se realizo de forma mecánica utilizando aplicaciones informáticas, específicamente archivos en Word y Excel donde el personal del área de tesorería, almacena la información de algunas solicitudes realizadas. Mediante la revisión de estos archivos se pudieron tomar y almacenar los datos que posteriormente serán utilizados como la muestra para crear el diseño de la base de datos.

Procesamiento de la información El procesamiento de la información se realiza mediante la tabulación de datos, esta se realiza de forma mecánica empleando el uso de la computadora, estos datos se cargan directamente en un archivo creado en la aplicación ofimática.

D. Marco Legal y Normativo

Régimen de la Propiedad Intelectual A fin de comprender cabalmente el tema bajo análisis, se hace necesario sobre todo para aquellas personas no familiarizadas con cuestiones legales hacer un breve repaso del marco legal que se debe aplicar a los programas de computación.

Frecuentemente los Ingenieros de sistemas, Ingenieros informáticos o profesionales afines se ven en la tarea de diseñar bases de datos y desarrollar software para organizaciones que lo solicitan o con el objetivo de crear nuevas ofertas en el mercado de los sistemas de información; si bien el software y las bases de datos son productos derivados del trabajo de estas ingenierías, también son obras protegidas por la propiedad intelectual, más específicamente por el Derecho de Autor lo que debe llevar a su creador a pensar además de llevar a cabo una solución informática o una base de datos, en la forma de protección de su obra por la vía del derecho de autor. Esta es una consideración de algunos aspectos relacionados con la propiedad intelectual aplicada al software y a las bases de datos [6].

La Propiedad Intelectual es el nombre que recibe la protección legal sobre toda creación del talento o del ingenio humano, dentro del ámbito científico, literario, artístico, industrial o comercial.

La protección que la ley colombiana otorga al Derecho de Autor se realiza sobre todas las formas en que se puede expresar las ideas, no requiere ningún registro y perdura durante toda la vida del autor, más 80 años después de su muerte, después de lo cual pasa a ser de dominio público.

En el caso del Software, la legislación colombiana lo asimila a la escritura de una obra literaria, permitiendo que el código fuente de un programa esté cubierto por la ley de Derechos de Autor. (http://www.iered.org/miembros/ulises/representacion-ideas/Derechos-Autor/propiedad_intelectual_en_la_legislacin_colombiana.html).

Para tener una idea más amplia de lo que son los derechos de autor y las normas de aplicación en el estado colombiano, se ha explicado un poco más a fondo este tema en el capítulo 15 donde se encuentran los anexos.

14. RECURSOS Y PRESUPUESTOS

A. Recursos Físicos

Los recursos físicos, financieros, logísticos y bibliográficos serán asumidos por parte de los estudiantes José Julián Ruiz Correa y Carlos Alberto Huertas Suarez, se utilizarán los equipos de cómputo y software propio para todas y cada una de las fases propias y tendientes a la elaboración, modificación, conclusión y presentación del mismo.

B. Recurso Humano e Institucional

Los recursos que se van a utilizar implicarán funcionarios del área financiera y de sistemas, los docentes formadores de la Universidad Libre seccional Pereira, que serán los guías para la elaboración y puesta en marcha del programa para la elaboración de los recibos de pago.

Los datos bibliográficos serán consultados en los libros de Universidad, páginas de internet y libros referentes a los temas mencionados anteriormente.

Las personas involucradas en el proyecto serán las siguientes:

Asesor de Proyecto

Carlos Alberto Ocampo Sepúlveda
Ingeniero de sistemas – UTP
Especialista Auditoría en Sistemas
Docente Catedrático
Ingeniería de Sistemas
Universidad Libre Seccional Pereira

Colaborador

Juan Manuel Cárdenas R.
Ingeniero de Sistemas y Computación – UTP
Director de Programa - Ingeniería de Sistemas
Universidad Libre Seccional Pereira

Directivos Institucionales

Dora Patricia Ospina Parra
Asesora Profesional de Adecco - Tesorera
Universidad Libre Seccional Pereira

Claudia Piedrahita
Ingeniera de sistemas
Directora Registro y Control
Universidad Libre Seccional Pereira
Administradora Base de Datos SINU

C. Recursos Financieros y Presupuesto

TABLA DE PRESUPUESTO

Tabla 18. Presupuesto

Recurso Humano	$/Hr.	Nro. Hr.	TOTAL	FUENTE FINANCIADORA
Investigador	40.000	10	400.000	PROPIOS
Director	30.000	40	1.200.000	UNIVERSIDAD
Asesor	20.000	20	400.000	UNIVERSIDAD
Total Talento Humano	**90.000**	**70**	**$ 2.000.000**	
Compra o Alquiler de Maquinaria y Equipos	**Costo Unitario**	**Cantidad**	**TOTAL**	**FUENTE FINANCIADORA**
Equipo de Laboratorio	0	0	0	
Maquinaria	0	0	0	
Computadores	1.500.000	2	3.000.000	PROPIOS
Software	150.000	2	300.000	PROPIOS
Total Maquinaria y Equipo	**1.650.000**	**4**	**$ 3.300.000**	
Fungibles	**Costo Unitario**	**Cantidad**	**TOTAL**	**FUENTE FINANCIADORA**
Materiales de prueba	0	0	0	
Materiales del modelo	0	0	0	
Reactivos, Insumos y consumos	0	0	0	
Libros	100.000	1	100.000	PROPIOS
Papelería y otros	50.000	1	50.000	PROPIOS
Total Fungibles	**150.000**	**2**	**300.000**	**PROPIOS**
Servicios públicos	80.000	2	160.000	PROPIOS
Viajes	200.000	2	400.000	PROPIOS
Gastos de Representación	50.000	2	100.000	PROPIOS
Arrendamiento local	0	0	0	
Total Otros Gastos	**480.000**	**8**	**960.000**	**PROPIOS**
SUBTOTAL			960.000	PROPIOS
Imprevistos 2-6%			57.600	PROPIOS
COSTO TOTAL DEL PROYECTO			**$ 1.017.600**	**PROPIOS**

Tabla 19. Recursos Financieros

FUENTE	COSTO A CARGO	[%]
Tesista	$ 4.717.600	74.67
Universidad	$ 1.600.000	25.33
Empresa Soporte	0	
Costo Total Proyecto	**$ 6.317.600**	**100**

15. CRONOGRAMA

Tabla 20. Cronograma

CRONOGRAMA AÑO 2013																						
ACTIVIDADES	Marzo - Abril				Mayo - Junio				Julio				Agosto				Septiembre				Oct	
	Semanas																					
	1-2	3-4	1-2	3-4	1-2	3-4	1-2	3-4	1	2	3	4	1	2	3	4	1	2	3	4	1	2
Definición tema de investigación																						
Investigación preliminar de tema																						
Entrevistas y Trabajo de Campo																						
Diseño de información primaria																						
Elaboración de la propuesta																						
Aprobación de la propuesta																						
Elaboración del anteproyecto																						
Entrega anteproyecto																						
Corrección anteproyecto																						
Análisis																						
Diseño Software																						
Desarrollo Software																						
Informe final y Entrega																						
Sustentación Proyecto																						

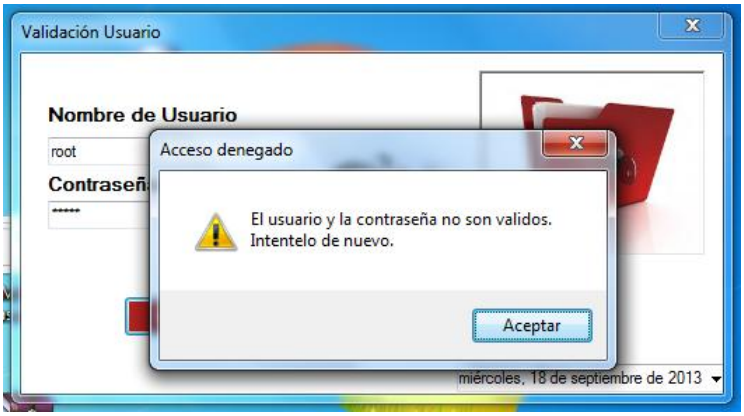
16. VALIDACIONES Y CONTROL

Se puede prevenir algunos errores en el ingreso de datos y mejorar el uso de una aplicación validando información mientras es ingresada a los campos de la aplicación.

Los controles de validación comprueban los datos proporcionados por el usuario en los formularios. La validación se produce cuando el formulario se envían y comprueban los datos proporcionados por el usuario y si estos no superan alguna de las pruebas de validación como el tipo de dato ingresado, si existe o no en la Base de Datos, si digitó caracteres no válidos, si su contraseña es errónea , entre otras; devuelve la página al dispositivo de usuario. Cuando esto ocurre, los controles de validación que detectaron los errores muestran una ventana emergente con el respectivo mensajes de error.

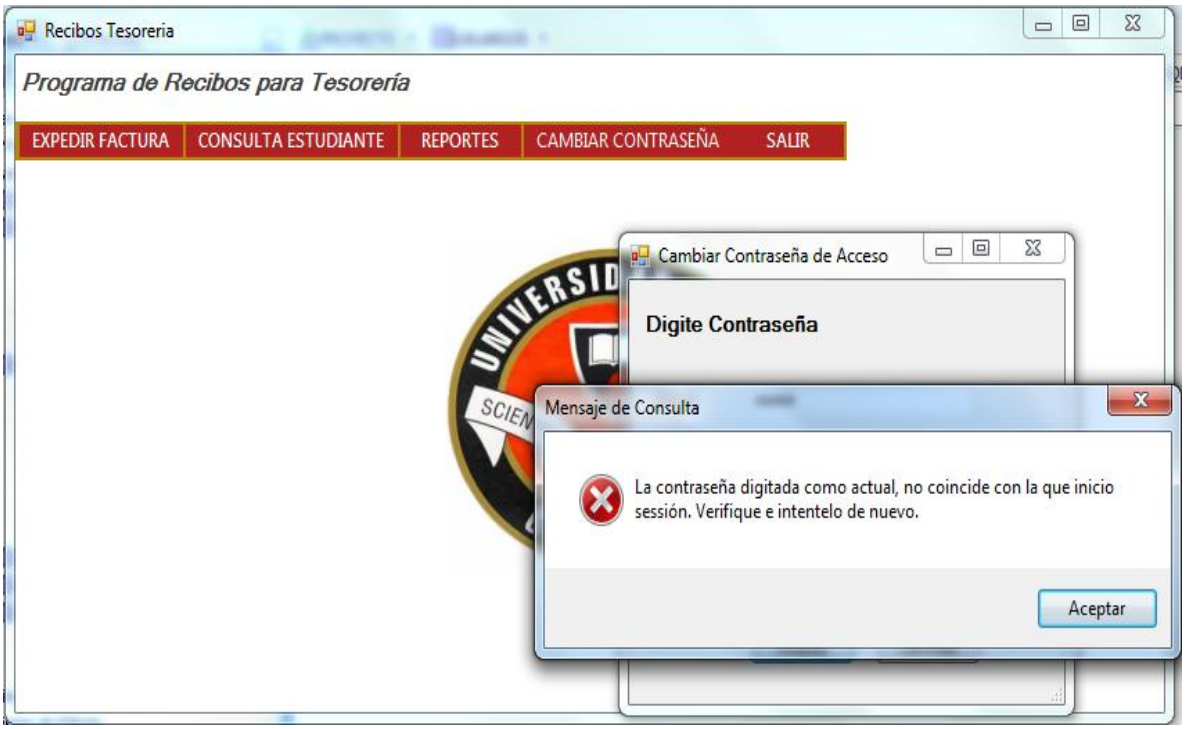
Se observará a continuación las ventanas más comunes de mensajes de error o de información de acuerdo a los ebventos que se presentan.

Ilustración 20. Validación de Usuario y Contraseña



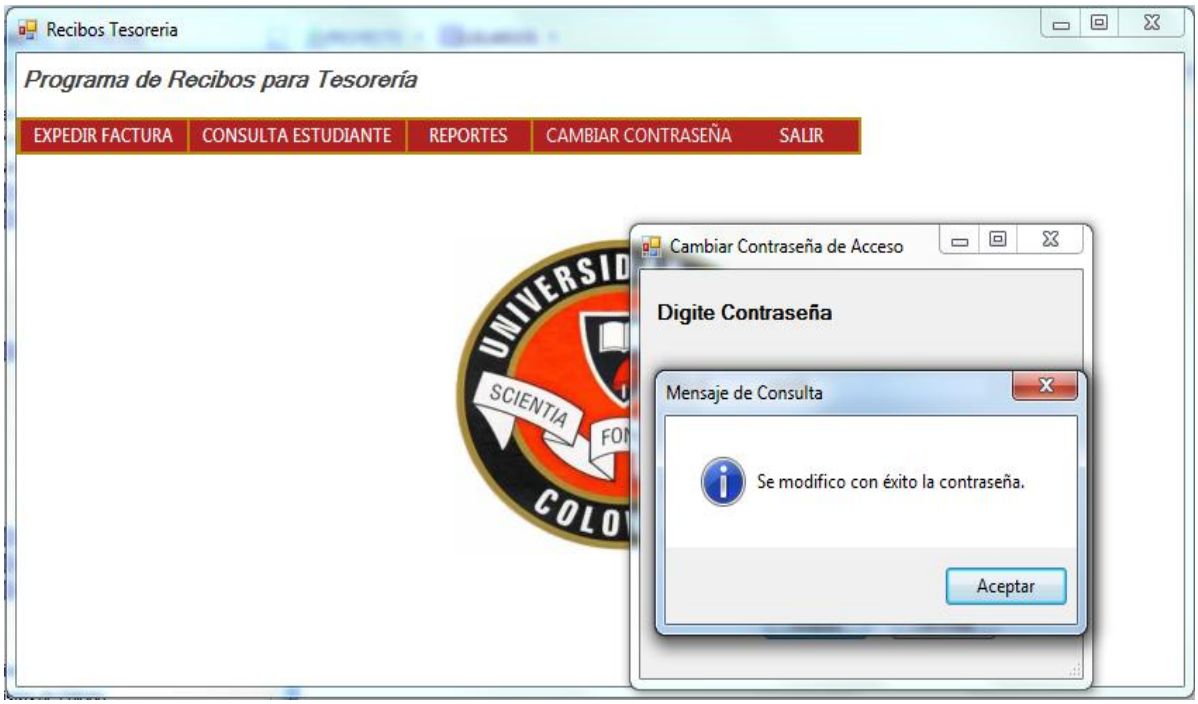
En la anterior validación, se presenta el caso en que un usuario con contraseña inválida trata de ingresar al sistema y no le es posible al no contar con validez de ingreso. La aplicación ha denegado el acceso y le presenta un mensaje de advertencia.

Ilustración 21. Validación de Cambio de Contraseña no Exitosa.



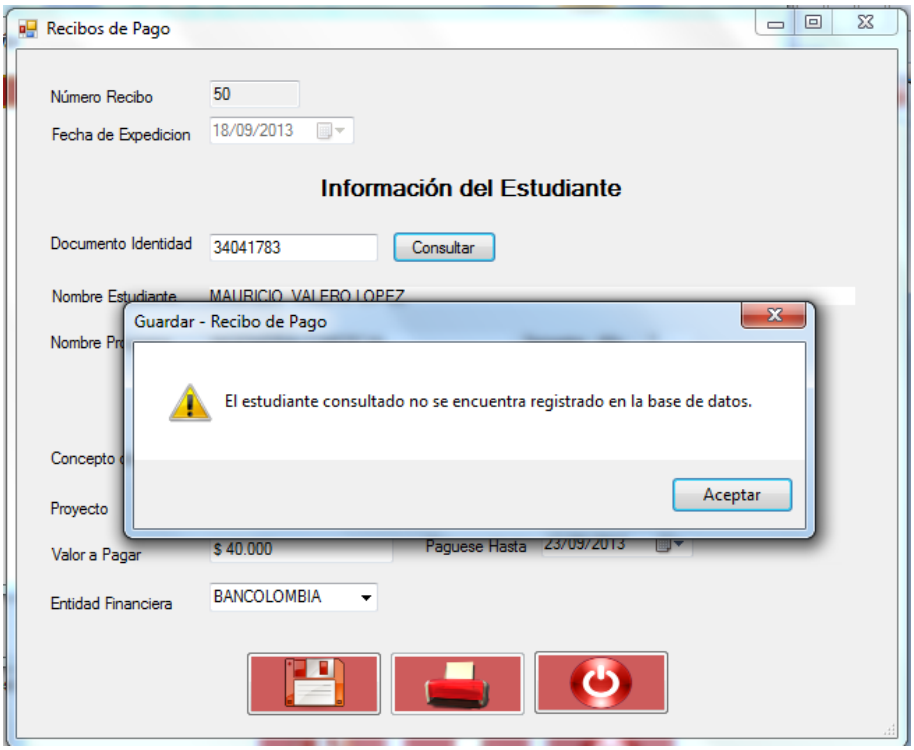
Aunque el usuario se validó correctamente en la pantalla de autenticación de usuario, ya en el menú principal ha intentado hacer una modificación de contraseña pero la clave inicial que ha digitado no coincide con la de autenticación y por esta razón la aplicación no le permite hacer cambio de contraseña, presentando el anterior mensaje de error. Este es un control muy importante ya que prohíbe que un usuario diferente al autenticado inicialmente pueda modificar la información de otro usuario.

Ilustración 22. Validación de Cambio de Contraseña Exitosa



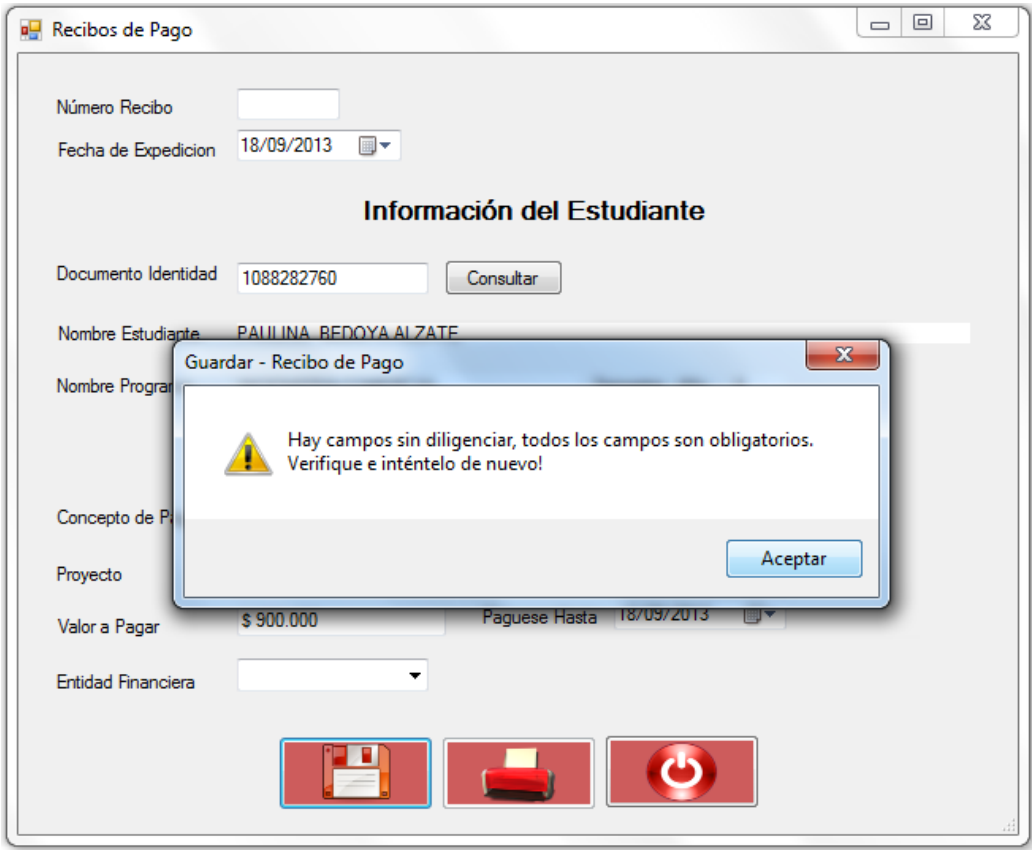
En este evento, el sistema ha permitido a un usuario hacer modificación de su contraseña, lo cual es necesario y por temas de seguridad se debe de forma periódica cambiar la contraseña a fin de evitar que una persona diferente pueda conocer y hacer uso de un usuario que no le corresponde.

Ilustración 23. Validación de Recibo de Pago. Estudiante no Existe en la Base de Datos



El anterior control, se presenta en el formulario de recibos de pago y valida que el usuario digite en el campo de Documento de Identidad, una número válido y que efectivamente exista en la base de datos; de lo contrario mostrará en ventana emergente la advertencia de que el estudiante que se quiere consultar no se encuentra registrado en el sistema de base de datos.

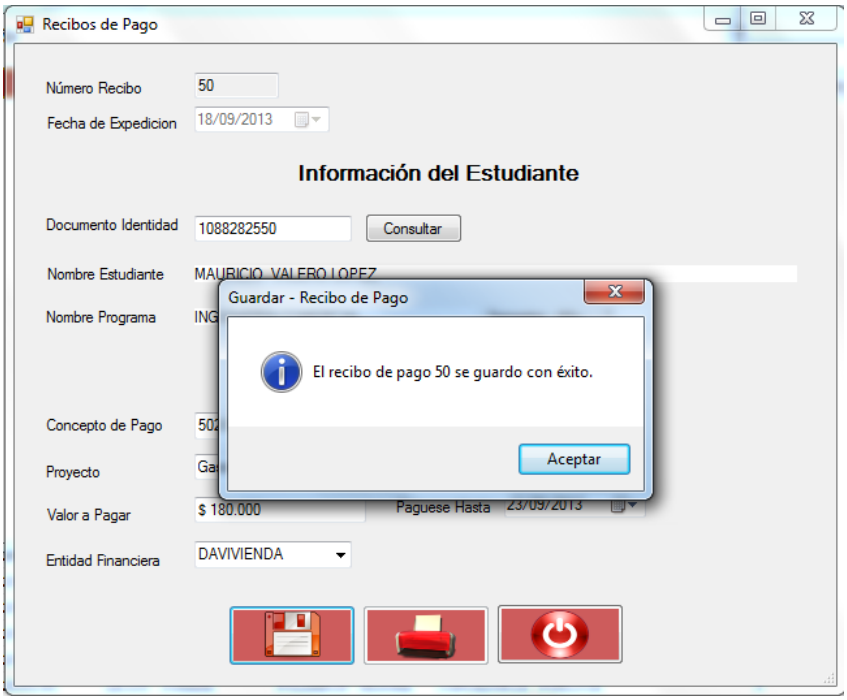
Ilustración 24. Validación de Recibo de Pago. Falta Campos por Diligenciar



En el formulario de recibo de pagos, todos los campos son de obligatorio diligenciamiento y si por algún motivo el usuario del sistema ha dejado un campo sin llenar, no podrá guardar el recibo porque se presentará la advertencia de que hay campos sin diligenciar, que debe verificar e intentar nuevamente.

Para el ejemplo anterior se puede observar como en el formulario no se ha elegido la entidad financiera para la cual se habrá de expedir el recibo de pago.

Ilustración 25. Validación de Recibo de Pago. Recibo Guardado Exitosamente



En esta validación del recibo de pago, el sistema muestra un mensaje informativo donde afirma que el recibo ha sido creado y almacenado exitosamente. Al hacer click sobre el mensaje aceptar ya se cuenta con la posibilidad de enviar la impresión de la factura.

17. CONCLUSIONES

- El conocimiento inicial que se tenga de un área específica a la que se pretende dar una solución en cuanto sus necesidades, son fundamentales para poder adelantar un proyecto de este tipo ya que de esta manera se garantizará el logro de los resultados que se esperan y no se desviarán los verdaderos objetivos por lo que es fundamental la investigación preliminar y la determinación de los requerimientos mediante la observación y entrevistas que se adelantaron a las personas directamente involucradas.

- La elección de la metodología UML que se utilizó fue la más adecuada para alcanzar la meta propuesta debido a que es esta quien determinó un orden el tiempo de duración del proyecto. Es por esto que para la realización de la aplicación para la elaboración de recibos de pago para el área de Tesorería de la Universidad Libre se aplicó el modelo en cascada que como bien se observó en capítulos anteriores, permite realizar cada una de las etapas secuencialmente alcanzando cada fase del ciclo de vida del software.

- El uso de modernas, sencillas y sólidas herramientas para desarrollo de Software como lo es Visual Studio .net, Oracle como base de datos; son determinantes al momento de construir el prototipo de la aplicación ya que se garantiza que se conseguirá la meta propuesta y que además permitirá que posteriormente se puedan hacer mejoras y modificaciones de acuerdo a las necesidades del usuario, garantizando además su ejecución en el Software y Hardware de dominio común y uso comercial con las cuales cuenta el área de Tesorería de la Universidad.

- En el momento de desarrolló la aplicación fue bien importante tener muy en cuenta que sus módulos debían ser sencillos de usar y gráficos para que en el momento que se pretenda capacitar e implementar el programa, se cuente con la mejor disposición de los usuarios y se garantice que el acoplamiento a la nueva herramienta será exitoso.

- Las pruebas finales de la aplicación antes de su implantación son vitales ya que permiten darle un tratamiento apropiado a las fallas o errores que puedan presentarse y que en un momento anterior no se hubieren tenido en

cuenta y así hacer las correcciones necesarias antes de llevarlo a producción.

- La existencia de un manual de funcionamiento o también llamado de usuario, es necesario para que las futuras personas que quieran interactuar con la aplicación, tengan la capacidad de enfrentarse al sistema sin mayores dificultades. Por otra parte puede volverse un documento de consulta permanente ante las inquietudes que se puedan generar posteriores a la capacitación que se brinda al usuario.

- Crear y dejar documentado el Software mediante un manual técnico, es bien importante ya que permite al departamento de Sistemas proponer futuras mejoras, correcciones o actualizaciones sin mayores dificultades al dejar documentada la información relevante a la construcción del mismo.

- En la experiencia adquirida a lo largo de la ejecución del proyecto, se evidenciaron algunos problemas de seguridad y control en la utilización de la herramienta saliente toda vez que no presentaba nivel de protección alguno, y donde todos los funcionarios del área e inclusive ajenos, tendrían la desafortunada oportunidad si se lo propusieran de ingresar al sistema y expedir una factura a su antojo, pudiendo inclusive manipular cifras de pago no autorizadas.

- Ha sido una experiencia enriquecedora haber tenido el compromiso de crear una herramienta que aunque es sencilla, permitirá que al ser implementada en la Universidad donde tuvimos la oportunidad de obtener nuestro título profesional, pueda resolver una dificultad que actualmente se presenta en la dependencia de Tesorería.

18. RECOMENDACIONES

- Se recomienda al área de Tesorería la instalación de la aplicación mínimo en dos máquinas de la dependencia ya que puede suceder que si una máquina falla, la otra pueda entrar a suplir la necesidad mientras se reinstala de nuevo.

- Se recomienda la buena conservación del back-up que se entregará en un CD, por si en algún momento sucede un evento inesperado con los computadores, se pueda tomar la herramienta e instalarla nuevamente con facilidad.

- Se recomienda la ejecución habitual de copias actualizadas de la información que se tenga almacenada en la base de datos del sistema de información con el fin de evitar las menores pérdidas por posibles fallas que se puedan presentar en la aplicación. Es recomendable realizar esta copia de manera mensual.

- Realizar constantemente los mantenimientos al Hardware tanto donde se ejecuta la aplicación, como a los demás equipos que colaboran con el área como Impresoras, Scanner, Ups, Redes eléctricas entre otras; vitales para el buen desempeño de las actividades propias de la Tesorería de la Universidad.

- Se recomienda al área de sistemas considerar la instalación del Software que se entrega, para solucionar las deficiencias presentadas en la tesorería en cuanto a la elaboración de recibos de pago.

19. BIBLIOGRAFIA

- Guía para la presentación de propuestas y anteproyectos para proyectos de grado e investigación. Universidad Libre. Facultad de Ingeniería.

- RODRIGUEZ, Gómez, G. y otros. Metodología de la investigación Cualitativa. Málaga: (1996). 145p.

- RUIZ, Olabuénaga, J. I. Metodología de la investigación cualitativa. Bilbao: (1996). 184p.

- SENN, James A. Análisis y Diseño de Sistemas de Información. Segunda Edición. (1992). 230p.

- Kendal & Kendall, Kenneth y Julie. Análisis y Diseño de Sistemas. Tecera Edición (1997). 190p.

- WINBLAND, Ann L. Edwards, Samuel D. King, David R. Software Orientado a Objetos. Primera Edición. (1993). 245p.

- [1] MICROSOFT VISUAL STUDIO 2012. {En línea}. {Consultado septiembre 2013}. Disponible en: http://es.wikipedia.org/wiki/Microsoft_Visual_Studio

- [2] VISUAL BASIC. {En línea}. {Consultado agosto 2013}. Disponible en: http://es.wikipedia.org/wiki/Visual_Basic_.NET

- [3] POPKIN, Modelado de Sistemas con UML. Software and Systems. (2001). 220p.

- HARMON, Paul. WATSON, Mark. Entendiendo UML. La guía del desarrollador. (1998). 250p.

- MORILLO, Franklin. Metodología orientada a objetos. Fases y clases. {en línea}. {05 de septiembre de 2013}. Disponible en: (http://franklinmorillouba.blogspot.com/2010/11/metodologia-orientada-objetos-fases-y.html).

- [4] BRITO, Gómez. Alexandra del Valle. Oracle. {En línea}. {15 agosto de 2013). Disponible en: (www.monografías.com/trabajos25/oracle/oracle.shtml.)

- [5] MICROSOFT VISUAL STUDIO. {en línea}. {20 de agosto 2013}. Disponible en: (http://www.microsoft.com/visualstudio/esn/products/visual-studio-overview)

- SICILIA, Miguel Ángel. De La Morena, Verónica. ¿Qué es la Reingeniería del Software. {en línea}. {10 de agosto 2013}. Disponible en: (http://cnx.org/content/m17438/latest/)

- [6] PELAEZ, V., Luis Eduardo. Buitrago C., Carlos Alberto. El derecho de autor aplicado en Colombia a los programas de ordenador (software) y las bases de datos. {en línea}. {10 de agosto de 2013}. Disponible en: (http://sanmartinbaq.edu.co/revistas/index.php/gd/article/download/42/pdf.).

- DIRECCION NACIONAL DE DERECHOS DE AUTOR. {en línea}. {15 de agosto de 2013}. Disponible en: (http://www.derechodeautor.gov.co/web/guest/home)

- [7] UNIVERSIDAD NACIONAL DE COLOMBIA. Preguntas frecuentes – Vicerrectoría de Investigación. {en línea}. {01 Junio 2013}. Disponible en: (http://www.viceinvestigacion.unal.edu.co/VRI/index.php?option=com_content&view=article&id=10&Itemid=101).

- PEÑA, Ayala. Alejandro. Ingeniería del software. {en línea}. {15 de julio de 2013}. Disponible en: (http://www.wolnm.org/apa/articulos/Ingenieria_Software.pdf)

- CODIGO DE BARRAS. {en línea}. {01 de septiembre de 2013}. Disponible en: (http://es.wikipedia.org/wiki/C%C3%B3digo_de_barras).

- [8] GS1 COLOMBIA. Código de Barras. {en línea}. {01 de septiembre de 2013}. Disponible en: (http://www.gs1co.org/nosotros/gs1colombia.aspx).

- [9]LA INGENIERIA DEL SOFTWARE. {en línea}. {01 de agosto de 2013}. Disponible en: (http://definicion.de/ingenieria-de-software/#ixzz2fe8L5xWZ).

20. ANEXOS

20.1 ¿QUE ES EL DERECHO DE AUTOR?

Es la protección que le otorga el Estado al creador de las obras literarias o artísticas desde el momento de su creación y por un tiempo determinado.

- **Algunos de los convenios internacionales sobre derecho de autor y derechos conexos ratificados por Colombia**

Acuerdo de Caracas de 1911 sobre Derechos de Autor entre Colombia, Bolivia, Ecuador, Perú y Venezuela, al cual adhirió Colombia mediante la Ley 65 de 1913.

Convención sobre Propiedad Literaria y Artística (IV Conferencia Internacional Americana, Buenos Aires, 1910), a la cual adhirió Colombia mediante la Ley 7 de 1936.

Convención Interamericana sobre Derechos de Autor en Obras Literarias, Científicas y Artísticas, firmado en Washington en 1946, al cual adhirió Colombia mediante la Ley 6 de 1970.

Convención Universal sobre el Derecho de Autor, firmada en Ginebra en 1952 y revisada en París en 1971, al cual adhirió Colombia por medio de la Ley 48 de 1975.

Convenio que establece la Organización Mundial de Propiedad Intelectual (OMPI) suscrito en Estocolmo en 1967, al cual adhirió Colombia mediante la Ley 46 de 1979.

Convenio de Berna para la Protección de las Obras Literarias y Artísticas de 1886, cuya última modificación se firmó en París en 1971, al cual adhirió Colombia a través de la Ley 33 de 1987.

Acuerdo por el que se establece la Organización Mundial de Comercio (OMC), que contiene el Acuerdo sobre los aspectos de los Derechos de Propiedad Intelectual

relacionados con el Comercio (ADPIC), al cual adhirió Colombia mediante la Ley 170 de 1994.

Tratado OMPI sobre Derecho de Autor, suscrito en Ginebra en 1996, al cual adhirió Colombia mediante la Ley 565 de 2000.

20.2 SOFTWARE Y BASES DE DATOS

- **Software**

RAE (Real Academia Española, http://www.rae.es) define el software como Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora. La Ingeniería de sistemas no entra en disputa con esta definición, pese a tantas que se leen en libros, documentos, etc. Se coincide con que técnicamente es un conjunto de instrucciones que permiten a la computadora llevar a cabo tareas determinadas. Por su parte la propiedad intelectual como disciplina del derecho atiende a sus definiciones y conceptos para precisar la forma de proteger el software bajo la modalidad del derecho de Autor. Con el término software se pretende describir la secuencia ordenada de instrucciones destinadas a ser asimiladas por un computador, a fin de lograr un resultado específico.

La Dirección Nacional de Derecho de Autor en la circular 05 del 9 de octubre de 2001 preciso que dentro del proceso de creación de un programa de computador, los desarrolladores generan en primera instancia un método algorítmico que servirá como estructura del programa final; éste se traslada a un lenguaje especializado (Cobol, Pascal, Visual Basic, Visual C, Oracle, Java, etc.), para constituir finalmente lo que se denomina código fuente. En este punto el programa no puede ser ejecutado por el computador, a este fin es necesario un procedimiento especial que transforme el lenguaje de programación a uno descifrable por la máquina, una vez terminado este proceso se entiende generado el código ejecutable. La concreción del código fuente es precedida por un proceso de orden intelectual, el cual en su gran mayoría queda sustentado de manera escrita, razón por la cual las diferentes legislaciones lo han asimilado a una obra literaria y, por ende, el régimen legal de este tipo de propiedad intelectual ha sido asignado al derecho de autor. Bajo este entendido, se concede protección al autor

del programa desde el momento mismo que crea su obra, sin que para ello requiera cumplir con formalidad jurídica alguna.

- **Protección legal del Software**

El entorno digital ha generado muchos inconvenientes a la propiedad intelectual, toda vez que el derecho no alcanza a regular los cambios y avances en dicho entorno, debido a la velocidad de los avances y progresos que dejan atrás la legislación vigente en la materia. Algunas corrientes han tratado de buscar protección del software a través de patentes, sin embargo, la OMPI (Organización Mundial de la Propiedad Intelectual) ha sido muy clara en sus apreciaciones y ha considerado que este debe ser protegido por la vía del derecho de autor, pues no es considerado como una invención que pueda ser sujeta de protección diferente, en este caso patentes. Es así como en Colombia a partir de la Ley 23 de 1982 se empieza a regular los aspectos legales relacionados con el software y a partir de esta regulación se crean nuevas normas y algunas circulares de la DNDA (Dirección Nacional del Derecho de Autor) reglamentando y ajustando la protección jurídica del software; también la Comunidad Andina de Naciones (a la que pertenece Colombia) promulgó el régimen común sobre derechos de autor y derechos conexos por medio de la Decisión 351 de 1992. A continuación extractos de la ley 23 de 1982, la decisión 351 de 1993 y algunas resoluciones y directivas de OMPI, DNDA en un resumen de lo que en materia de protección de software se ha logrado:

La existencia del derecho de autor atiende a la necesidad de reconocimiento de la creación intelectual y a criterios de índole económico que se traducen en beneficios tangibles para sus autores y/o quienes están autorizados para explotar dicha creación. La protección jurídica del software se concede al autor del mismo desde el momento en que se crea la obra, sin que para ello requiera cumplir con formalidad jurídica alguna, es decir, sin que deba adelantar algún trámite especial frente a alguna institución. La tendencia actual mundialmente aceptada respecto al software es que este está protegido por el Derecho de Autor en los mismos términos de las obras del dominio literario, pues reúne las características para ser objeto de dicha protección, tales como la originalidad, que se manifiesta como la concreción y materialización de una idea, y puede ser reproducida y definida por cualquier medio conocido o por conocer. La protección que se le otorga al autor del software tiene dos connotaciones con respecto a los derechos: morales y patrimoniales. Lo derechos morales son perpetuos, intransferibles e irrenunciables y facultan al autor para reivindicar en cualquier tiempo la paternidad de su obra;

oponerse a toda deformación o modificación que perjudique su honor o reputación o demerite la obra; estos derechos facultan también al Autor a publicar su obra o a conservarla inédita; a modificarla y a retirarla de circulación (artículos 11 Decisión Andina 351 y 30 de la Ley 23 de 1982).

En segundo lugar se encuentran los derechos patrimoniales, que constituyen una facultad exclusiva para realizar, autorizar o prohibir cualquier utilización que se quiera hacer de la obra, como la reproducción, la comunicación pública, la distribución pública, la importación y la traducción, adaptación, arreglo u otra transformación de la obra (artículos 13 de la Decisión Andina 351 y 12 de la Ley 23 de 1982). Los derechos patrimoniales están limitados en el tiempo y sobre estos se hará una descripción detallada más adelante. Los derechos patrimoniales son susceptibles de comercialización. En razón de lo anterior, cualquier persona que pretenda utilizar una creación protegida, deberá contar, salvo las excepciones legales, con la autorización previa y expresa del autor, de sus derechohabientes o de los titulares de los derechos patrimoniales en el caso de tratarse de una obra sobre la cual operó la transferencia de los mismos. Sin la licencia o consentimiento previo para la utilización de la obra podría llegar a ser calificada como ilícita, o como se conoce comúnmente en el caso del software como piratería, por vulnerar derechos sobre la creación protegida, siendo viable la aplicación de sanciones de tipo civil y penal.

- **Derechos que adquiere el Autor del Software**

El Autor del software adquiere derechos desde el momento en que inicia la creación del mismo, es decir, como se había expresado anteriormente, no se requiere ningún trámite adicional o especial para que el creado empiece a gozar de protección de su programa de computador. El creador del software tendrá el derecho exclusivo de realizar o de autorizar cualquiera de los actos siguientes:

- Reproducir la obra: copiar o autorizar la copia de la misma en cualquier medio análogo o digital.
- Efectuar una traducción, una adaptación, un arreglo o cualquier otra transformación de la obra, y
- Comunicar la obra al público mediante la representación, ejecución, radiodifusión o por cualquier otro medio; o para el caso del software la puesta en venta en sitios comerciales.

- Licenciar el software, según lo considere por máquina, individual, corporativa, etc. y sin perjuicio de las excepciones autorizadas por la misma ley.
- Los derechos en materia de propiedad intelectual corresponden al autor durante su vida, y después de su fallecimiento disfrutarán de ellos sus causahabientes o quienes legítimamente los hayan adquirido, por el término de ochenta años [6].

Ahora bien cuando se ha desarrollado un producto de Software para la Universidad en calidad de estudiante como requisito de grado debe tenerse en cuenta los siguientes interrogantes:

- **¿Quién se apropia de los derechos de autor sobre los trabajos de grado, trabajos finales y tesis?**

Por regla general es el estudiante quien ostenta la titularidad de los derechos morales y patrimoniales de autor sobre su trabajo de grado, tesis o trabajo final. En algunas excepciones los derechos patrimoniales de autor no los ostenta el estudiante, cuando se ceden los derechos patrimoniales a favor de la Universidad o de un tercero, en virtud de proyectos de investigación o extensión, a partir de los cuales se desarrolla el trabajo de grado respectivo. En cualquier caso, el estudiante conserva sus derechos morales sobre el trabajo, y en tal sentido la paternidad sobre la obra debe serle reconocida. En tal sentido lo estipula el artículo 24 del Acuerdo 35 de 2003 del Consejo Académico, así:

La calidad de autor sobre la obra literaria y/o artística que constituye el documento final de los trabajos de grado y tesis la detenta el estudiante. Cuando el trabajo de grado o la tesis del estudiante se realice dentro de un proyecto de investigación o extensión financiado por la Universidad o por una entidad externa o por ambas, será necesario que la Universidad establezca previa y expresamente mediante contrato debidamente suscrito por los autores y las partes, las condiciones de producción de la obra, las contraprestaciones correspondientes y la titularidad de los derechos patrimoniales.

La propiedad intelectual sobre los productos, informaciones, resultados diseños o datos útiles y susceptibles de ser protegidos como propiedad industrial corresponderá a la Universidad y/o al financiador, según contrato previa y debidamente suscrito con los estudiantes, el cual podrá incluir cláusulas de

manejo confidencial de la información usada y alcanzada. De ninguna manera esa condición deberá constituirse en obstáculo para la publicación del trabajo de grado o la tesis.

En concordancia con lo anterior, la Dirección Nacional de Derechos de Autor, mediante Circular No 06 del 15 de abril de 2002, manifestó lo siguiente:
El derecho de autor es un reconocimiento que el Estado hace a los autores, a través de la Constitución y la Ley, respecto de sus obras literarias y artísticas, al entregarles instrumentos que les permiten reivindicar su condición de titulares sobre las mismas.

Estos derechos surgen en favor del autor sin considerar el fin con cual fue creada la obra, siendo además irrelevante la calidad del creador, es decir, la ley no distingue si es un estudiante, un profesor o un investigador, así como tampoco es preciso establecer dónde tuvo lugar la creación o el tiempo que se haya utilizado, a efectos de esa misma protección.

Así, los derechos de autor sobre una obra literaria o artística, como lo sería un trabajo de grado, son de la persona que la realizó, quien la elaboró imprimiendo todo su ingenio e inteligencia. Es su expresión la que queda plasmada en lo producido, siendo por lo tanto el titular de los derechos morales y patrimoniales de la creación. En consecuencia, si la obra es realizada por un estudiante, será él, a la luz de la legislación vigente en materia de derecho de autor, el titular de todas las prerrogativas y facultades que la misma concede.

- **¿El Director del Trabajo de grado debe ser considerado como autor del mismo?**

El Director del trabajo de grado o tesis, no es el autor del trabajo de grado o tesis, sino como se indicó anteriormente lo es el estudiante quien en últimas ha desarrollado su labor literaria o científica, bajo la orientación y tutela del Director.

Sobre el particular, es importante tener en cuenta lo manifestado por la Dirección Nacional de Derechos de Autor en la circular antes reseñada, así:

El director de un trabajo de grado es por lo general un profesor de la institución de educación superior, a quien ésta le encomienda la tarea de brindar orientaciones o recomendaciones a uno o más estudiantes, quienes pretendiendo optar por su

título profesional deben preparar un escrito o una expresión artística como un plano, una maqueta, una pintura, una composición musical, un audiovisual, etc.. Su labor se concreta a señalar parámetros o líneas de investigación que inspiren al estudiante a fin de preparar finalmente su trabajo de grado. De tal forma, se considera como autor de la obra a la persona que expresó y plasmó sus ideas mediante dicho trabajo.

En consideración a ello, el autor único y exclusivo será el estudiante que organizó, recaudó y plasmó toda la información recopilada, incluidas las directrices e ideas planteadas por el director; así, cuando éste proporciona y presenta diferentes opciones al estudiante o corrige dicho trabajo, no hace otra cosa que cumplir con una obligación que le ha encomendado la institución de educación superior a la cual pertenece, sin realizar ninguna expresión literaria o artística.

Es menester señalar que el artículo 6º de la Ley 23 de 1982 y 7º de la Decisión Andina 351 de 1993, consagran el principio universal de la no protección de las ideas. Por lo tanto, aún cuando el director realiza una valiosa labor de apoyo al aportar ideas, dicha contribución no está protegida por el derecho de autor.

- **¿En qué consiste una obra derivada?**

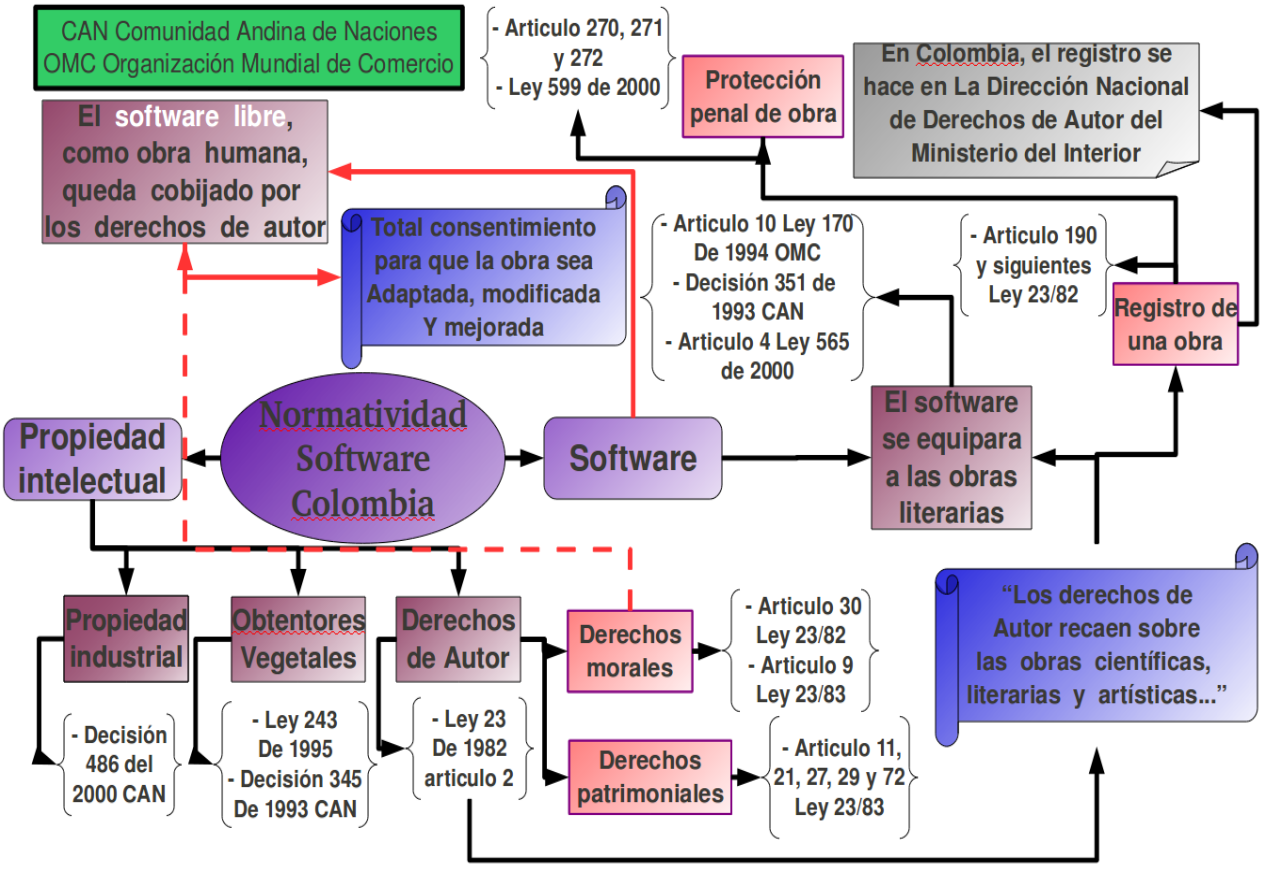
El literal j) del artículo 8 de la Ley 23 de 1982, define la obra derivada como aquella que resulte de la adaptación, traducción u otra transformación de una obra originaria, siempre que constituya una creación autónoma;. Dicha definición, se reitera en el artículo 12 del Acuerdo 35 de 2003 del Consejo Académico.

Si como producto del trabajo de grado, pasantía o tesis, se genera una obra derivada, es decir, una creación autónoma que tiene por origen lo planteado en dicho trabajo, los derechos de autor sobre la obra derivada recaen en los que participaron en la elaboración de la misma, en concordancia con lo dispuesto en el inciso segundo del artículo 14 del Acuerdo 35 de 2003 del Consejo Académico y la Ley 23 de 1982.

En ese orden de ideas, tal como lo establece el parágrafo único del artículo 24 del Acuerdo 35 de 2003 del Consejo Académico, si del trabajo de grado, pasantía o tesis, se desprende otra obra, los autores de la obra serán quienes participaron en la elaboración de la misma, en otra palabras, si en la obra derivada participa el docente que anteriormente dirigió el trabajo de grado, este tiene derecho a figurar

como autor en la obra derivada, situación que se deriva de su condición de autor (es decir de creador que plasma en un lenguaje su obra), más no de su condición de director del trabajo de grado o tesis respectiva [7].

Ilustración 26. Normatividad Colombiana respecto al Software



20.3 CARTA DE LOS TESISTAS DIRIGIDA AL COMITÉ DE PROYECTOS

Pereira, Julio 09 de 2013

Señores:

COMITÉ DE PROYECTOS DE GRADO

Universidad Libre Seccional Pereira

Ciudad

Asunto: **Presentación Proyecto de Grado**

Estimados Señores:

Atentamente nos dirigimos a ustedes con el fin de presentar el Anteproyecto de Grado, para optar al título de Ingeniero en Sistemas.

Agradecemos de antemano la atención prestada y la pronta respuesta a esta solicitud.

Cordialmente;

JOSE JULIÁN RUIZ CORREA	**CARLOS ALBERTO HUERTAS SUAREZ**
C.C. 10.006.698	C.C. 10.002.234
Estudiante Ingeniería en Sistemas	Estudiante Ingeniería en Sistemas

21. PRUEBAS AL PROTOTIPO DE LA APLICACIÓN

21.1 INFORMACIÓN GENERAL DE LA APLICACIÓN

Nombre: Aplicación para la elaboración de recibos de pago área tesorería universidad Libre seccional Pereira por conceptos no establecidos en Siul

Versión: 1.0.0

Estado: En desarrollo

Desarrolladores: José Julián Ruiz Correa

Carlos Alberto Huertas Suárez

Grupo de pruebas: José Julián Ruiz Correa

Carlos Alberto Huertas Suárez

21.2 METODOLOGÍA DE LAS PRUEBAS

Con el fin de verificar la funcionalidad de la aplicación, se realizó el diseño de las pruebas teniendo en cuenta la evaluación de diferentes criterios entre los que se encuentran: las entradas, la salida esperada y la salida obtenida de cada uno de los módulos que se tiene bajo revisión. Cada caso de prueba estará conformado de la siguiente manera.

Ilustración 27. Formato del Caso de Prueba

NOMBRE	TIPO
NÚMERO DE PRUEBA	No. Prueba "XX"
MÓDULO	"Nombre del módulo"
ACCIÓN	"Objetivo del Módulo"
DESCRIPCIÓN DE LOS DATOS DE ENTRADA	"Datos Ingresados"
SALIDA ESPERADA	"Resultado que espera"
SALIDA OBTENIDA	"Resultado que obtiene"
ETADO DE LA PRUEBA	"Satisfactoria" "Errada"

Ilustración 28. Diseño del Caso de Prueba 01. Expedir Factura

NOMBRE	TIPO
NÚMERO DE PRUEBA	No. Prueba 01
MÓDULO	Expedir Factura
ACCIÓN	Expedir factura de pago al Estudiante para ser cancelada en una entidad Financiera
DESCRIPCIÓN DE LOS DATOS DE ENTRADA	Usuario debe ingresar a la aplicación documento de identidad del estudiante para que este traiga los datos del mismo. De igual manera se digita el concepto de pago, el banco y si hará cambio de valor a uno de los dos ítem que lo permiten y finalmente Guardar el documento.
SALIDA ESPERADA	Formulario sea guardado exitósamente y al imprimir salga la factura impresa.
SALIDA OBTENIDA	Efectivamente el sistema guarda con éxito la factura mostrando un mensaje informativo
ETADO DE LA PRUEBA	**Satisfactoria**

El anterior formato (Ilustración 28), valida la correcta expedición de una factura de pago y su posterior impresión. Su resultado es exitoso.

Ilustración 29. Diseño del Caso de Prueba 02. Consulta Estudiante

NOMBRE	TIPO
NÚMERO DE PRUEBA	No. Prueba 02
MÓDULO	Consulta Estudiante
ACCIÓN	Consulta todos los recibos que han sido expedidos a un estudiante en particular, en un rango de tiempo predefinido por el Usuario del Sistema
DESCRIPCIÓN DE LOS DATOS DE ENTRADA	Usuario debe ingresar a la aplicación documento de identidad del estudiante para que este traiga los datos del mismo. De igual manera se digita la fecha inicial y la fecha final de consulta
SALIDA ESPERADA	Al hacer click sobre el botón Consultar, muestre el listado indicado y si hace click sobre el botón imprimir, se genere un reporte para impresión
SALIDA OBTENIDA	Efectivamente el sistema trae la consulta indicada y si se envía su impresión se genera el formato para tal fin.
ETADO DE LA PRUEBA	**Satisfactoria**

El anterior formato (Ilustración 29), valida la correcta consulta de un estudiante y los recibos de pago que le han sido expedidos en un tiempo establecido. Su resultado es exitoso.

Ilustración 30. Diseño del Caso de Prueba 03. Reporte Histórico

NOMBRE	TIPO
NÚMERO DE PRUEBA	No. Prueba 03
MÓDULO	Reporte. Histórico
ACCIÓN	Consulta todos los recibos que han sido expedidos mediante la aplicación en un rango de tiempo predefinido por el Usuario del Sistema
DESCRIPCIÓN DE LOS DATOS DE ENTRADA	Usuario debe ingresar a la aplicación la fecha inicial y la fecha final de consulta
SALIDA ESPERADA	Al hacer click sobre el botón consultar, muestre el listado indicado y si hace click sobre el botón imprimir, se genere un reporte para impresión
SALIDA OBTENIDA	Efectivamente el sistema trae la consulta indicada y si se envía su impresión se genera el formato para tal fin.
ETADO DE LA PRUEBA	**Satisfactoria**

El anterior formato (Ilustración 30), valida la correcta consulta de los recibos de pago que se han expedido a través de la aplicación en un tiempo establecido. Su resultado es exitoso.

Ilustración 31. Diseño del Caso de Prueba 04. Reporte Concepto Pago

NOMBRE	TIPO
NÚMERO DE PRUEBA	No. Prueba 04
MÓDULO	Reporte. Concepto Pago
ACCIÓN	Consulta todos los recibos que han sido expedidos mediante la aplicación por un concepto en específico y en un rango de tiempo predefinido por el Usuario del Sistema
DESCRIPCIÓN DE LOS DATOS DE ENTRADA	Usuario debe ingresar a la aplicación el tipo de Concepto a consultar y la fecha inicial y la fecha final de consulta
SALIDA ESPERADA	Al hacer click sobre el botón consultar, muestre el listado indicado y si hace click sobre el botón imprimir, se genere un reporte para impresión
SALIDA OBTENIDA	Efectivamente el sistema trae la consulta indicada y si se envía su impresión se genera el formato para tal fin.
ETADO DE LA PRUEBA	**Satisfactoria**

El anterior formato (Ilustración 31), valida la correcta consulta de los recibos que han sido expedidos de un concepto en particular y en un tiempo establecido. Su resultado es exitoso.

Ilustración 32. Diseño del Caso de Prueba 05. Cambiar Contraseña

NOMBRE	TIPO
NÚMERO DE PRUEBA	No. Prueba 05
MÓDULO	Cambiar Contraseña
ACCIÓN	Hacer el cambio de la contraseña de un usuario
DESCRIPCIÓN DE LOS DATOS DE ENTRADA	Usuario debe ingresar a la aplicación por medio del botón "Cambiar Contraseña" y en el formulario desplegado digitar su contraseña actual; posteriormente debe ingresar una nueva contraseña, confirmación de la misma y dar click en el botón Aceptar para que el cambio sea efectivo. Posteriormente debe abandonar la Aplicación y volver a ingresar para que compruebe el cambio de su contraseña
SALIDA ESPERADA	Que la nueva contraseña valide el acceso al sistema
SALIDA OBTENIDA	Efectivamente el sistema validó el acceso con la contraseña modificada
ETADO DE LA PRUEBA	**Satisfactoria**

El anterior formato (Ilustración 32), valida el correcto cambio de la contraseña de un usuario previamente validado en la aplicación. Su resultado es exitoso.

Ilustración 33. Diseño del Caso de Prueba 06. Salir

NOMBRE	TIPO
NÚMERO DE PRUEBA	No. Prueba 06
MÓDULO	Salir
ACCIÓN	Hacer click sobre el botón para abandonar la aplicación
DESCRIPCIÓN DE LOS DATOS DE ENTRADA	Usuario debe hacer click sobre este botón y el sistema se debe cerrar
SALIDA ESPERADA	Que la aplicación se cierre completamente
SALIDA OBTENIDA	Efectivamente la aplicación se cierra
ETADO DE LA PRUEBA	**Satisfactoria**

Con esta última prueba se valida que tan pronto se haga clic sobre el botón Salir, la aplicación efectivamente se cierre.